Situation Assessment for Mobile Robots

Beck, Anders Billesø

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

Technical University of Denmark

Anders Billesø Beck

Situation Assessment for Mobile Robots

PhD thesis, September 2012

DTU Electrical Engineering
Department of Electrical Engineering

DANISH TECHNOLOGICAL INSTITUTE

This thesis is submitted to the Department of Electrical Engineering, at the Technical University of Denmark in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

The work was performed within the IndustrialPhD program in collaboration between the Danish Technological Institute (DTI), Robot Technology and the department of Automation and Control, DTU Electrical Engineering in the period from July 2009 to September 2012. The project was partially funded by the Danish Agency for Science, Technology and Innovation.

Supervisors of the project have been Associate Professor Ole Ravn and Associate Professor Nils Axel Andersen from Automation and Control, DTU Electical Engineering.

Industrial supervisor was Claus Risager, ph.d., head of Danish Technological Institute, Robot Technology.

Abstract

Mobile robots have become a mature technology. The first cable guided logistics robots were introduced in the industry almost 60 years ago, but in this time the market has only experienced a very modest growth. However, considering the achievements in research the last 15 years within perception and operation in natural environments together with the reductions of costs in modern sensor systems, the growth potential for mobile robot applications are enormous.

Many new technological components are available to move the limits of commercial mobile robot applications, but a key hindrance is *reliability*. Natural environments are complex and dynamic, and thus the risk of robots misinterpreting the environment or failing to detect critical circumstances is unavoidable. To deal with this challenge, the control of robot applications must be able to handle imperfect observations and gracefully recover from unavoidable errors. The controller needs to know what is going on.

This thesis addresses exactly this problem from the hypothesis that an assessment of the situation for the robot, will be able to contribute with essential knowledge to the robot control and enable the understanding of the current situation as well as predict the future status.

A novel framework for situation modeling are presented, which applies an Extensible Markov Model (EMM) to represent the spatio-temporal nature of situations. On-line data-streams from the robot sensors and algorithms are processed using stream-based clustering to build the spatio-temporal structure or match the situation of the robot to existing states. Situation prediction is proposed using an on-line graph-search of maximum likelihoods in the EMM.

The developed software modules are integrated in a new software architecture, which facilitates integration into any robotic control framework and uses on-line visualization of the spatio-temporal graphs to optimize situation classification.

The results are evaluated in three real-world scenarios, which successfully evaluates capabilities of the proposed situation assessment framework within *detection of known spatio-temporal relations*, *deviation from known spatio-temporal patterns*, and *detection of known critical situations*.

Resumé

Mobile robotter er blevet en moden teknologi, hvor de første kabelførte logistikroboter blev introduceret i industrien for nœsten 60 år siden. Igennem denne tid, har markedet for mobile robotter i industrien dog kun bevidnet en meget moderat vœkst. Forskningen inden for mobile robotter har dermod bevidnet en målrettet indsats de sidste 15 år, som har resulteret i stœrke resultater inden for algoritmer og styring. Disse resultater, samt, at nødvœndige advancerede sensor systemer er faldet markant i pris, åbner et enmormt vœkstpotientiale inden for mobile robotter.

Mange nye teknologi-komponenter er allerede klar til at flytte grœnserne for anvendelsen af mobile robotter i naturlige omgivelser, men en nøgleudfordring for at dette kan lykkes er deres *pålidelighed*. Naturlige omgivelser er komplekse og dynamiske, hvilket medfører der er en uundgåelig risiko for at robotten misfortolker omgivelserne eller forfejler detektionen af kritiske omstœndigheder. For at håndtere disse udfordringer, må styringen af robot-applikationen vœre i stand til at håndtere fejlbehœftede observationer og komme videre efter uundgåelige fejl. Robottens styringssystem er nødt til at vide hvad der egentligt foregår.

Denne afhandling adresserer netop dette problem, ud fra den hypotese, at vurdering og forståelse af robottens situation kan bidrage med essentiel viden til robotstyringen og gøre denne i stand til at forstå robottens nuvœrende situation samt forudse hvad der vil ske i den nœre fremtid.

Et nyt system for situations modellering bliver prœsenteret, der anvender en Extensible Markov Model til at representere den spatio-temporale natur af situationer. On-line data-streams fra robottens sensorer og algorithmer bliver behandlet ved brug af stream-baseret clustering til at bygge den spatio-temporale model og til at matche robottens situation til eksisterende tilstande. Forudsigelse af situationer er realiseret ved brug af on-line grafsøgning af maximum likelihood i den spatio-temporale model.

De udviklede software moduler er integreret i en software arkitektur, som muliggør integration i et hvilkårligt robotstyrings system og anvender on-line visualizering af den spatio-temporale graf for at optimere klassifikation af situationer.

Resultaterne er evalueret i tre brugsscenarier der, med success, vurderer situation assessment systemet i forhold til *detektion af kendte spatio-temporale relationer*, *afvigelser fra kendte spatio-temporale mønstre* og *detektion af kendte kritiske situationer*.

Acknowledgments

I would first of all like to thank my university supervisors Ole Ravn and Nils A. Andersen for the cooperation, advice and encouragement. We have had a long journey together, and I appreciate the knowledge, the inspiration and the friendship that I bring with me.

I secondly want to express my gratitude to my industry supervisor and manager Claus Risager, for taking me onboard for this project and for your endless inspiration, motivation and energy in this project, and in many to come.

I would also like to thank the other members of the Automation group at DTU for inspiring discussions, in particular my office-mate Søren Hansen and robotics-codewizard Christian Andersen for good and inspirational times. A special thanks to Dorthe Hansen for patient and thorough proofreading of this thesis.

Thanks also to my colleagues at DTI for the daily support, discussions and challenges. In particular my team leader Kurt Nielsen for sparring, encouragement and gentle pushes. Further thanks to Mikkel Viager for the collaboration and assistance in the experimental work and to Lars Dalgaard and Bridget Hallam providing valuable feedback to this thesis.

A thanks to the Robotics and Embedded Systems department of the Technische Universität München for welcoming me at the department and giving me a pleasant and educational stay. Special thanks to Dr. Reinhard Lafrenz for inviting and supervising me, to Prof. Alois Knoll for facilitating my stay and finally to Alexander Perzylo for enjoyable office companionship and the endless hospitality of you and your family.

Thanks to my friends and family for always being there with support and motivation, this journey would not have been the same without you. A grateful thought is sent to my father Gunnar who is not with us to witness this day, but whom without I would not have been the person that I am, and achieved what I have today.

Finally, my greatest appreciation and gratitude is to my wife Tanja and my two sons Frederik and Mads, without your love, support and energy this would not have been possible. Thank you for bearing with me throughout this journey, and in particular the past months.

Anders Billesø Beck

September 2012

Contents

1

Introduction

Mobile robots are becoming a mature technology and applied in an increasing number of domains and applications. Mobile robots have existed in industry for almost 60 years as Automated Guided Vehicles (AGVs), but their distribution has only been increasing at a modest pace. Most new applications and products that emerge are merely evolutions of the original AGV principles using improved sensor technologies and better integration into production IT infra structures. In recent years, where most obvious processes in industry have become automated using specialized production equipment and industrial manipulation robots, introduction of mobile robots for improving logistics performance are the next logical step. Nevertheless, the market for AGVs are still orders of magnitude smaller, where IFRs World Robotics 2012 statistics note 2.100 sold systems worldwide in 2011 compared to 181.000 sold units of industrial manipulation robots (IFR Statistical Department, 2012).

There is currently a diffusion of mobile robot application away from industrial settings towards service robot applications, both in professional and domestic domains. New technologies and significant drops in sensor prices begin to enable mobile robots to enter new and more challenging applications. Unfortunately, these new robots have a difficult route to the commercial markets. The leading applications in relation to sales are military purposes for professional services (where Unmanned Aerial Vehicles account for 75 % of the sales) and the domestic service area is mostly dominated by household robotics such as vacuum cleaners, lawn mowers and entertainment robots (IFR Statistical Department, 2012). New applications like professional cleaning, inspection, maintenance, and security robots only hold a neglect able part of the sales figures.

A major impact area for mobile robots in recent years are hospitals. Hospitals have enormous volumes of intra logistics of medical supplies, food, clean and dirty linen, blood samples, and surgery equipment. The cost of this logistics can amount to more than 30 % of the annual hospital budgets, which makes the introduction of mobile robots for logistics obvious in this domain. A number of companies have established them in this domain, and have developed platforms which navigate on natural environment contours and are safe to operate among humans. In Denmark, experiments with mobile robots have already been conducted at Svendborg Sygehus in 2009 for transportation of blood samples. Led

by the Danish Technological Institute, a larger scale integration of the Aethon[1] robot platform will commence in the fall of 2012 in Sygehus Sønderjylland, initially also for blood samples but later for several other applications such as custom food delivery.

The growth potential for the mobile robot market is considered very large, as the basic technologies are mature and many new applications and technologies are investigated in this domain. The widespread market penetration of mobile robots has now been predicted almost continuously for the last decade, e.g. by Gates (2007), but nothing really seems to happen. Why is that?

1.1 Challenges for mobile Robots

Demonstrations in research laboratories demonstrate capabilities which significantly exceed what we find applied in commercial products. Mobile robotics have been subject to intensive research the last 15 years and a broad part of the key scientific challenges, such as localization, mapping, navigation planning and environment perception, are now considered solved.

Sensor systems have dropped significantly in price recent years, and new technologies emerge, which present previously unseen price/performance relationships, like the Microsoft Kinect cameras, which enable 3D perception for less than 80 euro. Together with cost effective mobile platforms, development systems can be set-up for very reasonable cost and pave the way for competitive commercial products.

One key obstacle of why so few commercially attractive products enter the markets despite of the technical possibilities is reliability. Reliable perception of natural environments, navigation among natural obstacles of any size and in any place, and detection of objects with only partially known geometries are really large challenges which yet have not seen any solutions that are fail safe. When integrating these challenges into one application, it sets a very high demand to the error handling capabilities of application control systems, which are an area where a limited amount of research results has been presented.

To make a reliable system for application control, it is essential to have a proper comprehension of the current situation to make informed decisions to recover from errors or, even better, to predict the likelihood of future errors in order to avoid them instead of recovering from them. The comprehension of situations for mobile robots and the prediction of future events are the issues studied in this thesis.

[1] Aetheon, Automated Hospital Delivery and Asset Management Solutions, http://www.aethon.com, 2012

1.2 Working hypothesis

The work presented in this thesis has been conducted in the context of an IndustrialPhD project at the Danish Technological Institute (DTI), Robot Technology. The three year program is co-funded by the Danish Agency for Science, Technology and Innovation and has the purpose to connect Danish industry and academia.

For DTI the goal of the project has been twofold as 1) to investigate into the challenges and applications for the future of mobile robotics and as 2) to specifically address the issue of reliability in robotics applications and intelligent interpretation of both errors and normal situations. The context throughout the work has been confined to mobile robotics, but with a general outlook in mind to ensure the possibility of future transfer to other domains.

The challenges of this thesis have focused on comprehending the current situation of the robot, and assessing if that complies with state estimates within the robot control, in order to identify possible critical situations and enable the robot control to handle them gracefully.

The approach to this thesis has been formulated in the two hypotheses, as it is assumed:

- that scenarios of application for mobile robots can be adequately described in a finite number of situations.
- that a generalized model for situation description and comprehension can be developed, which will be able to cover the majority of situations that will occur for a mobile robot in any area of application.

As an approach to this challenge, this thesis presents an investigation into the domain of situation assessment from a mobile robot application perspective.

1.3 Situation Assessment

Knowledge and comprehension of the current situation are defined in literature as situation awareness (Endsley, 1995), where the process of achieving, acquiring and maintaining situation awareness is known as situation assessment. Research and analysis into situation awareness for human operators dates back to World War I, where it was used to describe crucial performance characteristics for crews in military aircrafts. Later it has been used to generally describe the assessment capabilities of military performers and in the 1990s the term was adopted by the human factors area of research.

In the human factors domain, situation awareness (SA) has been investigated in complex dynamic decision making environments, such as air traffic control, operation of large automation systems (e.g. nuclear power plants) (Endsley, 2006), as well as ground fire commanders (Klein, 1989), in order to understand the processes of (SA) and how to design systems which optimize operator SA in the best possible way.

Situation assessment is also addressed in the domain of information fusion, as a process of achieving higher level knowledge by fusion of a number of lover level data sources. Strategies in this domain are based on development of relationships between objects and events in the context of their environments. Application of research in this level of information fusion is found in the military domain for battlefield overviews or intelligence gartering, or in the civilian domain for systems for cyber security.

The presented approach to situation assessment considers the contributions from both areas of research to form a model for situation assessment and investigates as the first the use of situation assessment in the domain of mobile robot applications.

1.4 Contributions

By the work of this thesis, a number of contributions was made, which can be divided into three main categories:

1. Situation Assessment

The work presented is a novel approach towards a connection between situation assessment from an information perspective and the challenges of mobile robotics. The challenges of mobile robot applications are investigated and a number of popular open source robotics frameworks have been investigated to map their capabilities in higher level application control. Only very limited capabilities are found in this area, and those frameworks which have presented decision architecture and included higher level tools are now abandoned from future development. The frameworks that have caught popularity are the architecture neutral, which are easy to use, but leaves the entire application development to the user.

Modeling and representation approaches to situations are investigated and define situations as spatio-temporal relations between objects. The analysis lead to the proposal of three strategies to achieve situation assessment in the mobile robot domain: *detection of known spatio-temporal relations*, *deviation from known spatio-temporal patterns*, and *detection of known critical situations*. These strategies are used to form the situation model and addressed specifically in the experimental evaluation.

The DTU MobotWare framework is discussed and presented as robotics framework for the work of this thesis. The architecture and approaches to high-level application control of DTU MobotWare have been published in:

- A. B. Beck, N. A. Andersen, J. C. Andersen, and O. Ravn. Mobotware - a plug-in based software framework for mobile robots. In *7th IFAC Symposium on Intelligent Autonomous Vehicles*, Lecce, Italy, 2010.
- A. B. Beck, N. A. Andersen, and O. Ravn. Mission management for Mobile Robots. In L. Akdahl, editor, *The fourth Swedish Workshop on Autonomous Robots*, volume I, Vesterås, September 2009. Robotdalen.

2. Modeling framework

A novel situation modeling framework is presented, which can parameterize and represent situations in a general context and has been evaluated in a mobile robot context with good results.

The model accepts heterogeneous data from a flexible data-source system which creates a good infrastructure to any robotics framework. Situations are represented in the spatio-temporal Extensible Markov Model (EMM) framework by Dunham et al. (2004). The EMM enables a situation model to evolve on-line and yet to maintain the spatio-temporal structure. The Markov Chain representation of EMM is efficient for analyzing state relationships in both the spatial and temporal domain in order to comprehend the situations the relationships describe.

Real-time data are processed and matched to states in the EMM using stream-based clustering, as a non model-based approach to dealing with sensor noise and doing data reduction. A prediction engine enables on-line prediction of future events based on graph-search in the likelihoods captured in the spatio-temporal model.

The situation modeling approach and parts of the experimental results of this thesis have been published in:

- A. B. Beck, C. Risager, N. A. Andersen, and O. Ravn. Spatio-Temporal Situation Assessment for Mobile Robots. In *14th International Conference on Information Fusion*, Chicago, 2011.

3. Software architecture

A new software architecture is proposed to integrate the contributed software modules in a flexible, expandable and modular framework. The architecture is implemented in the Situation Assessment Platform (SA-Platform).

A flexible data source module enables seamless integration to any robotics framework or other sensor data source. The evolution of the spatio-temporal EMM is visualized in real-time by adaptation and real-time interface of the Gephi graph visualization software. The visualization proved very beneficial for human comprehension of situations represented by the EMM.

The use of the SA-Platform has enabled demonstration and evaluation of the situation modeling framework using on-line and real-world examples.

1.5 Thesis outline

The structure of this thesis follows the classical scientific model: *analysis*, *synthesis* and *evaluation*, where chapter 2 and 3 analyses the challenges and assumptions of the presented work, followed by chapters 4, 5 and 6 which synthesize and discuss the proposed theoretical approaches of the thesis. Finally, chapter 7 presents the experimental evaluation of the proposed approach.

The structure and outline of each chapter are therefore as follows:

Chapter 2 entitled *Mobile Robots* addresses the domain of application for mobile robots. The chapter also presents an analysis of a number of robotics software frameworks and reviews their capabilities of higher level modules for application control. Furthermore, DTU MobotWare is presented together with a discussion of the challenges in mobile robot task execution.

Chapter 3 entitled *Situations for Mobile Robots* first introduces the definition of situations and critical situations for mobile robots. This leads to a presentation of the strategies for situation assessment.

Chapter 4 entitled *Situation Assessment* presents the fundamental of situation assessment from an Information Fusion perspective as well as from the Human Factors perspective. The chapter furthermore presents the proposed spatio-temporal situation definition together with the situation modeling framework and prediction engine.

Chapter 5 entitled *Processing Information Streams* presents the challenge of processing on-line streams of information with focus on a non model-based approach. This leads to the design of the stream based clustering engine.

Chapter 6 entitled *Situation Assessment Platform* presents the architecture and design of the Situation Assessment Platform software framework.

Chapter 7 entitled *Experimental Evaluation* presents the experiments conducted to evaluate the performance of the situation modeling framework. A total of three experiments is conducted, where each experiment addresses one of the strategies for situation assessment.

Chapter 8 entitled *Conclusions* presents the final conclusions of this dissertation and highlights the major contributions as well as a brief discussion of future work.

2

Mobile Robots

In scientific and popular literature the emergence of second generation intelligent service robots outside laboratories has been predicted repeatedly for the last 10 years (Gates, 2007). In this decade advanced algorithms for localization and navigation have been maturing towards industrial strength implementations and the capabilities of affordable computing and sensor hardware continuously increase. Yet we still do not see intelligent service robots in our home, in our workplace or at the shopping centers. The reason is that reliable long term operation in dynamic and indeterministic environments is a huge challenge, not only for the navigation and localization systems but also for the system capabilities of handling unplanned and even unexpected events.

As of today, even simple service operations as grass cutting, floor cleaning and logistics such as retrieving and delivering items are carried out by humans because mobile service robots lack the reliability to deliver professional grade results. In opposition to industrial robots, mobile service robots must operate in environments, which are designed with no regards to the robots that must operate in them. Daily operating conditions involve frequent changes in the environment and a high likelihood of the robot getting caught in situations that standard navigation methods are unable to resolve or could bring the robot in undesirable or even potentially harmful situations.

In the following chapter I will present the application where mobile robots play an important role. I investigate how they are used in industry, in domestic settings and how they are demonstrated in state-of-the art research applications. The purpose of the investigation is to clarify which challenges mobile robots are facing and how applications are designed to handle them. Furthermore, I investigate how the efforts within community driven software frameworks for mobile robots address design and execution of mobile robot applications and what capabilities they have for handling potential errors.

DTU MobotWare is the mobile robot control software framework used for the work of this thesis. In the last part of this chapter the architecture of MobotWare is presented. Finally, I discuss strategies for handling errors in mobile robot applications and why situation assessment can be an important contribution to improving this capability.

2.1 Mobile Robots in General

Mobile robots distinguish themselves from their counterparts, the industrial manipulators, by being able to move about in their environment. When industrial manipulation robots work in the same environmental setting, most parts of the environment and workspace can be considered static.

As definition of this thesis I consider mobile robots as all types of robots, which are capable of moving about in their environment without any fixed attachment (no linear axes or rails), where robots referred to as industrial robots account for industrial manipulators who are mechanically fixed to the environment.

Many simplified approaches to mobile robot control also assume a static environment, but yet the robot has to move about in the environment. Just moving about introduces a higher level of uncertainty than for industrial robots, as there is no direct way of getting feedback to whether the commanded motions actually were executed as expected, wheels might be slipping. In most industrial robot settings, the environment can actually kept adequately unchanged, so that a simple control script with fixed coordinates will be enough to control an installation for years.

For mobile robots, the real challenge is that even in dedicated industrial settings it is very difficult to enforce a static environment, as human workers almost always will be present in the workspace of the mobile robots. Humans will move things around and intercept the operation space of the robot, forcing the robot to sense the environment in real-time and reconstruct plans, motions and estimates dynamically.

Workspaces for mobile robots are almost always much larger than for most industrial robot installations. In these large workspaces it is often difficult to determine accurate coordinates of objects in the workspace using a global coordinate system, often simply because of measurement inconvenience but also accuracy of affordable measurement tools. Besides of often operating with low accuracy information on elements in the environment, the technologies commonly applied to estimate the location of the robot location in the environment have an accuracy which is several orders of magnitude worse than what is known in industrial robots.

Mobile robots are especially suited for tasks where there is a need for a large workspace. The tasks are not only on the sense of pure logistics and inspection, but a lot of effort in research and development projects is currently put into mobile manipulation robots. When reach of the industrial manipulators become fairly large (> 2 m), the only available robots are often large scale heavy lifting manipulators. So despite fairly low payloads, i.e. 10 kg for packing, a robot with a payload of several hundreds of kilos is often used, just because it has the reach. These huge robots in light packaging stations can be replaced by significantly smaller robots on mobile bases.

2.2 Commercial applications for mobile robots

2.2.1 Industrial applications

Mobile robots gain an increasing popularity in industrial settings. Commercial systems are most commonly used in larger warehouse environments and production facilities that produce or consume large quantities of material. In these settings they transport material to and from storage to production machines and they transport products to packaging, around warehouses and onto trucks. These logistic functions require complex handling of orders and storage locations, which can be overwhelming for humans but fairly easily integrated into order and Warehouse Management Systems (WMS). The tight integration of WMS and task control of the mobile robot fleet entails that most suppliers of mobile robot systems deliver complete systems of warehouse, order and logistics management together with the mobile robot hardware.

Navigation and localization of the industrial platforms, known as Automatic Guided Vehicles (AGVs), were previously done by traditional methods, such as embedding magnetic/electromagnetic guide wire in factory floors or painting ultraviolet reflective tracks. Most recent installations use reflective laser guidance, where a rotating laser is mounted on AGVs well above human head level and reflective markers are placed in exposed places, such as corners. Having a geometric map of reflectance markers allows the laser system to triangulate the position of the AGV quite robustly. Newer systems, like the SICK NAV350 illustrated in figure 2.1, use a combination of marker based localization and distance measurements of the natural environment, so that environment contours can be used for localization where no reflective markers are visible. This allows laser guidance systems to be used, even in situations such as truck loading/unloading where no reflective markers can be used.

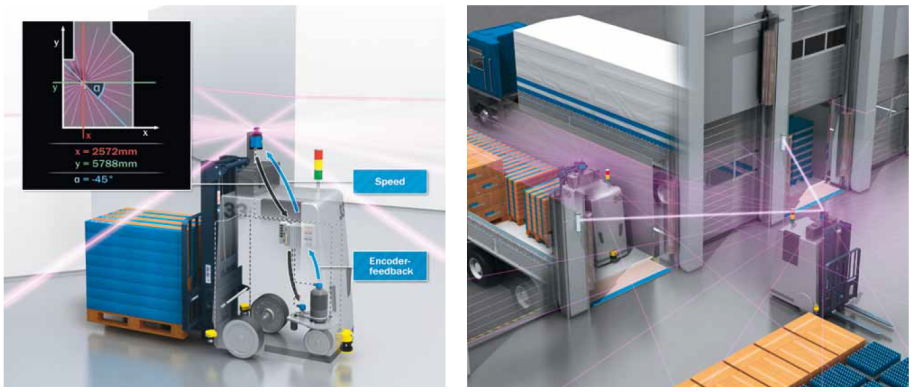

Figure 2.1: Loading and unloading of trucks using reflective markers and environment contours for localization can be accomplished by the new Sick NAV350 navigation system

Laser guidance has a fairly good accuracy, for example the SICK NAV3xx series has a position accuracy of ±4 mm with a systematic and statistical error of ±10 mm (SICK, 2012), which is on par or slightly better than most research grade localization systems. To distinguish laser guided robots from the

traditional AGVs, they are often referred to as Laser Guided Vehicles (LGVs). Another popular system for laser guided navigation is produced by Guidance Navigation[2].

Examples of industrial mobile robot systems

The following shows examples of commercial mobile robot systems for industrial applications, both illustrating the focus of industrial mobile robot products as well as the technologies which are applied to in the robotic platforms:

Elettric 80[3] is a leading Italian supplier of logistics and warehouse management solutions, especially for end-of-line tasks. For logistics and intra warehouse operation they exclusively use LGVs in many different forms to transport goods to and from packaging systems. Positioning in the Elettric 80 LGV platforms is done by the SICK NAV platform. Quite a lot of suppliers similar to Elettric 80 exist, such as Egemin, Frog, Atab, Savant, CoreCon, etc. who all supply traditional system infrastructures to transport pallets and other bulk containers around production and packaging facilities, as well as they supply complete WMS to manage the logistics processes and LGV fleets.

Figure 2.2: Commercial LGV platform from Elettric 80 for light warehouse automation. All rights of this image are courtesy of Elettric 80©.

Aethon[4] is an American based mobile robot developer, who provides innovative robotic hospital delivery systems and asset management solutions. More than 100 hospitals throughout USA employ customized Aethon solutions for intra logistics. Tasks like transporting blood samples, dirty laundry and warm meals can be automated using Aethons TUG mobile robot platform, illustrated in figure 2.3. The platform uses sensing in the natural environment for navigation, based on ultrasonic, infra-red and laser-scanners measure distances to the environment and navigate the robot. Maps for navigation are created from

[2]Guidance Navigation Limited, http://www.guidance.eu.com, 2012

[3]Elettric 80, Total Business Solutions, http://www.elettric80.com, 2012

[4]Aetheon, Automated Hospital Delivery and Asset Management Solutions, http://www.aethon.com, 2012

building CAD files or learned by the robot/operator if they are not available. Evaluation and integration projects of the TUG platform into Danish hospitals will commence in fall 2012 with DTI as lead integrator.

Figure 2.3: The Aethon TUG delivery platform, which will be demonstrated in Denmark in the fall of 2012.

Swisslog[5] is a swiss based globally leading integrator of logistic solutions. The focus of Swisslog is to provide solutions for the industry segments of food and beverage, retail, pharmaceuticals and healthcare. In the European research community, Swisslog is especially known for their TransCar mobile robot platform, which is employed at many European hospitals and illustrated in figure 2.4. The platform is also laser guided, using SICK safety laserscanners to localize and guide it through the hospital environment. The laserscanner is placed at floor level, which makes robust tracking of environment contours slightly challenging, as it has to look through human feet and other non-stationary obstacles. A benefit of having the navigation system at floor level is that it eliminates the need for an additional safety laserscanner to avoid collisions with humans and static objects.

[5]SWISSLOG, A leading global provider of integrated solutions, http://www.swisslog.com/, 2012

Figure 2.4: Swisslog TansCar logistic robot platform is designed to lift and move material carts around hospitals.

Kiva Systems[6] is a truly innovative player on the mobile robotics market. The Kiva Systems concept is a mobile robot system specifically designed for fulfilling orders in the large volume mail-order industry. The small Kiva mobile robot platforms transport racks in warehouses directly to the operators, who mix-pack cartons for shipment. In this way, the Kiva system automatically brings the right products directly to the operator, who can effectively pick the items and fulfill orders much more efficient than manually picking with a trolley around a large warehouse, as shown in figure 2.5. Navigation of the Kiva mobile robot platforms is done, using a rather simplistic approach by reading position bar codes on the floor, which are placed in a grid in the entire warehouse floor. Ultrasonic sensors are used for obstacle avoidance. Where Kiva especially distinguishes itself from competitors, is that their solution does not focus entirely on mobile robot technology but has rather focused on optimal flow of materials and using mobile robots as just one of the means of achieving that. In May 2012, Kiva Systems was acquired by Amazon for no less than $775 million, which underlined the impact the company gained in its short 9 year lifetime.

[6]Kiva Systems, A Different kind of Material Handling Company & Complete Warehouse Automation Solution, http://www.kivasystems.com, 2012

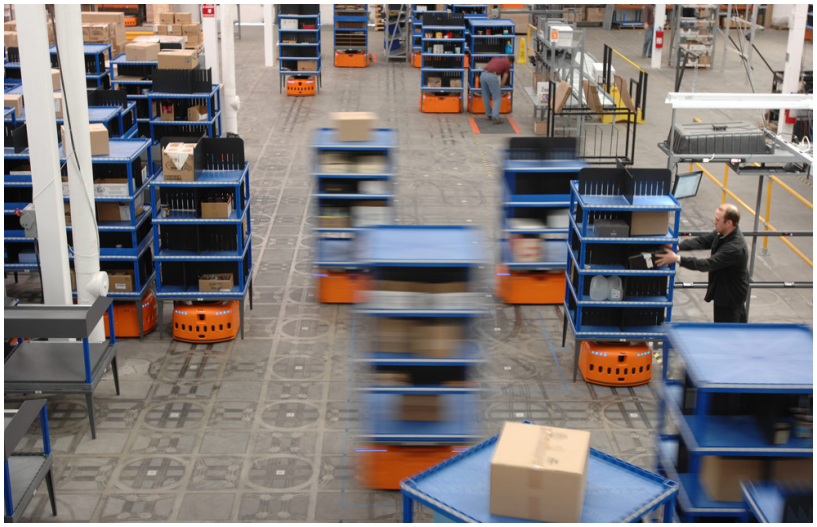

Figure 2.5: The Kiva Systems robotic platforms transport shelve units from warehouse inventory to packing stations in parallel, so operators always have the shelves and items at hand.

This section of industrial mobile robot usage makes it quite obvious, that mobile robots are used in many different domains in industrial settings. However, the largest majority of the robot systems are still designed with the sole purpose of transporting goods around in the respective facilities.

2.2.2 Non-industrial use of mobile robots

Domestic-like settings have been the primary target and playground for mobile robotics research for quite some time. The reason why research has focused a lot on domestic environments is that domestic settings cannot be customized for the robot systems in the same degree as in industrial settings, where the workspaces are significantly modified to suit the mobile robots. Mobile robots for domestic use must operate in the natural environments of humans.

Two different usage scenarios exist for non-industrial mobile robots: professional service and domestic service. Professional services range from healthcare and indoor cleaning and pool cleaning. Domestic services include vacuum cleaning robots, lawn moving robots and robots for telepresence. Revenue is not produced by professional or domestic services at the same level as industrial services, so earnings by employing mobile robots can often only be counted in the number of minutes which they liberate human service personnel, whereas industrial systems can improve earnings by much larger factors as they can improve production throughput.

Professional cleaning would seem like an obvious target for automation using mobile robots. Large scale floor washing machines already exist and only an addition of a navigation system and navigation of large open spaces are a specialty for mobile robots. Building commercial professional cleaning robots have been attempted by many large industrial players, such as Hefter (using Siemens control), Karcher, Subaru, Servus, Hako, Comac, Cybernetix, Robosoft

and many more. Examples of the robots can be seen in figure 2.6. Unfortunately, these robots had competed with very low salaries in the cleaning sector and large cleaning capabilities for the existing manual solutions. None of previously mentioned solutions had commercial success and can be bought today (Elkmann et al., 2009).

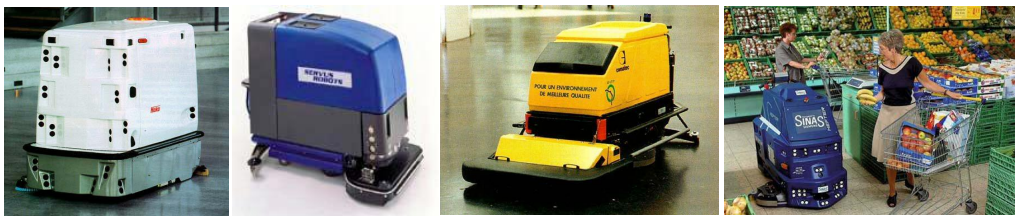

Figure 2.6: Professional floor cleaning solutions produced by Hako, Servus, Comac and Hefter. None of the solutions had commercial success and are available for sale today.

In the personal segment, vacuum cleaning robots have had major success and represent what most non-roboticists consider a mobile robot. The segment was pioneered by the American company iRobot, who initially started building robots for military purposes in 1990. In 2002 they introduced the Roomba, the first mobile robot vacuum cleaner to gain global popularity with a total of more than 5 million sold units. The Roomba has been updated on several occasions through its life cycle, but the upgrades have mostly been mechanical and on software, not on navigation system. In the following years, many vacuum robots have followed using principles very similar to the Roomba. Recently new technologies are finding its way into the vacuum cleaning robots, such as the Samsung Navibot who uses an upwards facing camera to map and navigate in rooms and follow straight cleaning paths. The latest contribution to vacuum robots is the Neato Robotics XV-15, who uses a proprietary low cost rotating laserscanner to map and navigate the cleaning area in similar manner to State-of-the-art research systems. The laserscanner allows the XV-15 to clean just without bumping into all obstacles, to clean in a very ordered fashion, structure the sequence of rooms for cleaning and suspend cleaning for charging and resume again from the spot where it stopped.

Figure 2.7: Development in the cost-sensitive market of personal vacuum cleaning robots is also beginning to introduce technologies previously only used in research. The Neato Robotics XV-15 uses a cost effective rotating laserscanner for mapping and localization, making it capable of navigating entire houses, vacuum in straight lines, and is finishing one room at the time.

2.3 State-of-the-art for mobile robot applications

Research within mobile robotics is pushing the limits of mobile robot applications even further. Sensor technologies are not limited by commercially viable business cases, and robustness does not necessarily need to meet industry standards of thousands of hours between failures.

The following section will look into state-of-the-art of mobile robot applications in order to further lay out the landscape of the challenges and situations which need to be tackled in these systems.

2.3.1 Autonomous Cars

A major research initiative from the American Defence Advanced Research Projects Agency (DARPA) has directed quite some attention towards development of autonomous cars. Starting with the 2004 Grand Challenge, DARPA posted a competition for autonomous cars. The 2004 and 2005 challenges were to navigate 240 km and 212 km respectively in the Mojave Desert. The Grand Challenge was superseded in 2007 by the Urban Challenge, a 96 km urban race at the closed George Air Force Base and including challenges such as obeying traffic regulations, negotiating traffic and obstacles as well as merging into traffic. Throughout the years of the Grand Challenge, the capabilities of the competing vehicles significantly improved performance, from 2004 where no vehicle passed 11 km to 2007 where 6 vehicles completed the challenging course. The rivalry of the competition was especially between Stanford, the winning team in 2005 and

second place in 2007, headed by Sebastian Thrun and Carnegie Mellon, second place in 2005 and winner in 2007

In the slipstream of the Grand Challenges, autonomous driving slipped a bit in the background of the robotics community. Some initiatives kept working on the challenge, such as the Urban Challenge 2007 competitor, VisLab department from the University of Pharma. Their recent prototype "BRAiVE" (figure 2.8), is a completely customized x-by-wire, where the sensory system of 10 cameras, 4 laserscanners, 16 single-point lasers, GPS and IMUs is completely integrated into the car. The primary goal of BRAiVE is to be a research platform to develop industry-grade capabilities of driver assistance, such as lane assistance, adaptive cruise control and elements of autonomous driving.

Figure 2.8: The BRAiVE autonomous car research platform.

Head of the Stanford Racing Team, Sebastian Thrun, joined Google on a sabbatical and co-invented Google Street View in this period. Inspired by the Grand Challenge and Street View, the cooperation between Google and Thrun took another level in 2010, when Google announced the Self-driving Car project headed by Thrun. Googles project brought renewed fuel to research in self-driving cars, and the achievements from their Toyota Prius, equipped with a VeloDyne LIDAR HDL-64E 64-beam laserscanner are impressive.

The accumulated distance travelled by the Google car has now surpassed 300.000 miles with an ambition to pass 1 million miles soon. In 2011 the state of Nevada passed a law on public licensing of autonomous vehicles. The law went into effect in March 1, 2012 and the Google Prius was the first car to pass the autonomous driving test and be awarded a street license and a red set of license plates, marked with the infinity symbol and the text "autonomous vehicle" (Paul, 2012).

2.3.2 Field Robots

Robots for field operation are another area, where mobile robots are beginning to play an important role. Many manufactures of heavy agricultural equipment are intensively developing auto-pilot systems for their vehicles.

The Danish company AgroCom is now a part of the Dutch agricultural equipment manufacturer Claas Agrosystems. AgroCom makes the stereo vision systems Agrocom Eye Drive for the Claas family of tractors. Using the Eye Drive system for precision farming, tractors can follow narrow trails between rows of crops for weeding or spraying, or it can follow lines of hay for bailing, or harvesters can follow edges of the standing crops for maximum utilization. Figure 2.9 shows a Claas Axion tractor with an Eye Drive stereo camera setup. Furthermore, this tractor can be autonomously controlled by the DTU MobotWare control framework, which is described in more detail in section 2.5.2. Safety regulations do not allow the tractors to operate completely autonomously yet, so there are still operators to control the throttle of the tractor, but steering is done autonomously.

Figure 2.9: Claas Axion tractor equipped with Eye Drive stereo vision system and autonomously controllable by DTU MobotWare.

Research in field robotics domain has always been different than research within domestic service robots. Key challenges in field robotics are navigating and operating in large natural outdoor environments, whereas Service Robotics is mostly about relatively smaller indoor man-made environments. Autonomous driving can be considered to fit between the two areas, with navigation in large outdoor environments but following man-made traffic structures and rules.

Groups of field robot research is often a bit a-typical compared to traditional robotics research, as many groups have specific domain knowledge rather than being a core robotics research group. Examples are the group of robotics research in the Institute of Chemical Engineering, Biotechnology and Environmental Technology at the University of Southern Denmark, or Wageningen University in the Netherlands who is a university focused on agriculture, forestry and horticulture, but still a key player in field robotics in Europe. Many similar examples are common in the field robotics environment, illustrating that

challenges in this environment are just as much in the field processes and heavy equipment as they are within robotics.

2.3.3 Service Robots

Robots for providing services are a less well defined area than e.g. industrial robots. Since 2007 a working group of ISO has been reviewing the ISO8373, which will include an official definition of service robots. Until then, the IFR [7] has adopted a tentative definition of service robots as

> *"A service robot is a robot which operates semi- or fully autonomously to perform services useful to the well-being of humans and equipment, excluding manufacturing operations."* (International Frederation of Robotics, 2012)

In research, service robots are most commonly addressed as mobile manipulation robots. These robot platforms are the closest real siblings of the household service robots, known from sci-fi movies. When combining manipulation arms and mobile platforms, the robots can move about to grasp and operate things in a normal domestic environment. But comparing the enormous global efforts put into this domain and the results achieved, there is yet a significant way to go, before we see the robots enter our everyday kitchens.

In 2006, Scott Hassan founded Willow Garage in Menlo Park, California, a company to accelerate research in non-military robotics and promote open source software development within robotics. Through the Personal Robotics Program (PR Program), Willow Garage set off to push the limits of what can be done with personal service robots. Building on the robot platform developed from professor Ken Salisbury's lab at Stanford (PR1), Willow Garage developed the PR2 (Personal Robot 2) platform. The PR2 is a research-centric design with completely controllable hardware architecture, massive computation and sensor suite, omni-directional mobile base and safe dual manipulation arms. In 2010 Willow Garage made a call for proposals to a PR2 beta program, where research institutions could apply for borrowing one of 11 PR2s for two years against contributing their research results to the open source robotics community.

One of the groups who was awarded a PR2 in the beta program was the IAS group of the Technische Universität München (TUM), led by Professor Michael Beetz. The capabilities of the PR2 suited the research activities of the IAS group, who is striving towards making cognition enabled service robots. Examples of their research activities are the use of integration of semantic knowledge into computer vision for scene classification and object recognition and the integration of knowledge processing and reasoning to automatically specify and create the action sequences required to solve household tasks. Figure 2.10 illustrates the PR2 making popcorn at an official demonstration at the IAS laboratory in München.

[7] International Federation of Robotics

Figure 2.10: The PR2 making popcorn at an official demonstration at TUM IAS laboratory.

The Fraunhofer Institute for Manufacturing Engineering and Automation (IPA) has through more than 14 years developed another state-of-the-art service robot platform: the Care-O-Bot. The Care-O-Bot was initially designed as a tour guide robot in 1998, but the second generation in 2002 introduced a manipulator to enable the use of the platform for mobile manipulation research. The third generation Care-O-Bot 3 was designed as a mobile butler robot, capable of performing manipulation tasks safely with a rear-mounted industrial manipulator and serving humans with a touch-screen equipped tray in front.

In the research project WiMi-Care, the Care-O-Bot 3 was demonstrated in the nursing home Parkheit Berg in Stuttgart, Germany (Fraunhofer IPA, 2011). In the demonstration the Care-O-Bot was used to assist the nursing staff by providing glasses of water to the residents of the facility. By registration of water consumption in databases together with active follow up by interaction with the residents, the Care-O-Bot could single out the residents which have not had enough to drink and offer them additional drinks. In order to accomplish the application, the Care-O-Bot was capable of grasping cups, filling them at a water dispenser and place them on the tray for serving. Figure 2.11 shows Care-O-Bot serving a cup of water to a nursing home resident.

Figure 2.11: The Care-O-Bot 3 serving water to the residents of the nursing home Parkheit Berg in Stuttgart, Germany.

All the presented novel applications of mobile robotics clearly illustrate, that there are significant challenges and massive research connected to any advances in the possible applications within this area. Furthermore, when the large number of advanced sensors and software components are integrated in an application, there is a fairly high likelihood of any of these modules failing at some unexpected point and thus causing an application error. This generally motivates the need of a general situation assessment tool in these applications to learn the natural behavior of the application and to comprehend and possibly even predict application errors.

2.4 Robotic Software Frameworks

In research, a number of software systems have been developed and published to encapsulate research achievements and to form an infrastructure for running mobile robot applications. Before the millennium it was commonplace for each research group to develop and maintain their own software infrastructures to control the mobile robots. It caused software to be rewritten again and again in research groups all over the world and countless hours was spent, especially on developing software infrastructures to support development, configuration and execution of the robot software modules.

The purpose of this section is to analyze the characteristics of the most common robotics frameworks used as development platform for state-of-the-art robot applications. The particular focus of the analysis is to investigate the strategies used to execute the robot applications and which capabilities exist to ensure that applications are correctly executed and the mission goals actually

are achieved.

Pioneered in the early 2000's by Gerkey et al. (2001) with the Player project and Montemerlo et al. (2003) with the CARMEN toolkit, a consensus started to grow in the robotics community that open source software and common frameworks were the way to join forces and reach new goals with research robotics. From this period, many research projects turned to the use of these frameworks, as they provided infrastructure, hardware abstraction and simulation capabilities. However, many established research groups already had much work invested in their own frameworks and the result was initially that most existing frameworks became open source. The popularity of open source robot-control frameworks also opened a lot of discussion of how to develop reusable robot software components, i.e. by initiatives like the Best Practices in Robotics (BRICS) project (Bischoff et al., 2010) and how frameworks and functionality were difficult to keep separated (Makarenko et al., 2007).

Initiated by the software requirements of the Personal Robotics Program and the PR2 robot platform, Willow Garage presented a general open source robotics framework to the global robotics research community. A clear part of the Willow Garage strategy was to unite the forces of the robotic research community, by designing a strong and feature rich framework for community driven open source development. So in 2008, the Robot Operating System: ROS (Quigley et al., 2009) was presented. The impact of introducing ROS in the robotics research community has been second to none.

2.4.1 Architectures vs. Frameworks

Despite the widespread adaptation of robotic software frameworks, the primary benefit of using a framework was initially that researchers could avoid writing hardware device drivers. Some functionality exists in the cores of most frameworks but it is not until the last few years that robust modules e.g. for localization were considered a standard. The results were that there were not produced significantly better results on higher level tasks when frameworks were used than from groups who used self-designed and monolithic control software.

This claim is supported by a larger survey by Kramer and Scheutz (2007), which analyses nine public domain frameworks in depth, not only on features but also on the specific goals of the frameworks, the practical usability and the scientific impact (by related publications). Many software platforms are difficult to include in the comparison, such as Cyberbotics Webots (although this is quite widely used) and Evolution Robotics' ERSP, as they are closed source commercial platforms. The interesting NASA platform CLARAty (Nesnas et al., 2006) has not been included either, as the platform was not released to public domain at the time of the analysis. In June 2007 NASA choose to go open source with CLARAty, which caused some excitement in the robotics community. But seemingly it has lost momentum again, and nothing has been published on CLARAty since 2008. Other frameworks not considered in the survey was Lego Mindstorms, as it was too specific for one platform, as well as Orocos (Bruyninckx, 2001) and YARP (Metta et al., 2006), as Kramer and

Scheutz (2007) set a minimum requirement that one cohesive application should have been developed in the framework at the time of writing. Interestingly both Orocos and YARP have survived and become somewhat popular frameworks the following years. The frameworks included into the analysis were ultimately nine: TeamBots[8], ARIA [9], Player/Stage (Gerkey et al., 2001), Pyro, CARMEN (Montemerlo et al., 2003), MissionLab (MacKenzie and Arkin, 1998), ADE (Andronache and Scheutz, 2004) and MARIE (Coté et al., 2008).

The analysis of Kramer and Scheutz (2007) evaluates frameworks based on the following categories of criteria:

1. Specification	Formalisms, methodologies, and design tools.
2. Platform support	Hardware, low-level interface and system support.
3. Infrastructure	Components and capabilities as part of the framework.
4. Implementation	Application control and components of the architecture.

The criteria forms a very in-depth analysis of the frameworks, as the quality of each top-level criterion is evaluated on basis of 4-9 sub-criteria. In the specification category only an around half of the frameworks have architectural primitives implemented to support organization of modules e.g. for control, behaviors and perception, but all of them are considered to be architecture neutral. Being architecture neutral can be considered a benefit for rapid application development, as there are no stringent doctrines which must be followed, but when applications grow, it is often the cause for anarchy. Furthermore, when no distinct architectures are followed, it becomes much more difficult to develop general modules for higher level functionality when there are no clear definitions on which role the higher level modules play in the architecture. The evaluation results for each top-level criteria are shown in table 2.1.

Framework	**Feature**				
	Specification	**Platform**	**Infrastructure**	**Implementation**	**Total**
TeamBots	83 %	17 %	0 %	10 %	28 %
ARIA	67 %	33 %	38 %	47 %	45 %
Player/Stage	50 %	83 %	56 %	57 %	59 %
Pyro	83 %	83 %	31 %	73 %	64 %
CARMEN	67 %	83 %	50 %	33 %	47 %
MissionLab	100 %	67 %	63 %	50 %	60 %
ADE	100 %	50 %	100 %	60 %	74 %
Miro	83 %	50 %	56 %	47 %	53 %
MARIE	83 %	100 %	63 %	77 %	76 %

Table 2.1: Comparison of feature scores for the evaluated robotic frameworks by Kramer and Scheutz (2007).

[8] TeamBots ®, http://www.teambots.org/, ©Tucker Balch, 2000
[9] ARIA, Mobile Robotics Inc., http://robots.mobilerobots.com/wiki/ARIA, 2012

ADE and MARIE earn the highest score in total features in table 2.1 on the preceding page with 74 % and 76 % respectively, closely followed by Pyro with 64 %, all caused by superior results in one or two specific categories. This underlines that most frameworks are created with one specific ideology where it performs rather well, but overall good scores are difficult and some scores can even be contradictory to the ideology of the framework.

Digging into the evaluation key-points, especially interesting criteria in the context of this thesis are the fault tolerance and higher level reasoning capabilities of the frameworks. The level of features supported in the frameworks will tell important details about well handling of errors and deviation from planned actions are handled. Fault tolerance is evaluated mostly on basis of separation between components, on basis of whether the failure of one component will crash the framework and does any mechanism aid to its recovery. Player/Stage, CARMEN, MissionLab, MARIE and Miro do support component isolation by the use of 3rd party middleware (IPC and ACE). These features will allow the framework to keep running despite of software failures, but there is no functionality on application level to ensure that the mission gets correctly executed.

Another criteria evaluates the capabilities of predefined components of the frameworks. Besides the classical localization, route planners and vision components, some frameworks also contain higher level reasoning components, such as rule-based interpreters in ADE, MARIE and MissionLab, generic task planners in MARIE and MissionLab, and learning modules in TeamBots, Pyro and MissionLab. These components can all be beneficial for developing higher level robustness in application execution. Rules-based systems and task planners can be used to handle deviations from plans and learning modules can be used to further adapt to changes in application assumptions. It is still up to the application developer to implement these functionalities using the supplied modules.

Framework	**Total Usability**	**Total Impact**
TeamBots	35 %	10 %
ARIA	53 %	10 %
Player/Stage	82 %	80 %
Pyro	88 %	15 %
CARMEN	59 %	25 %
MissionLab	71 %	35 %
ADE	76 %	30 %
Miro	29 %	30 %
MARIE	50 %	35 %

Table 2.2: Comparison of usability and impact results for the evaluated robotic frameworks by Kramer and Scheutz (2007).

Player / Stage received a general high usability score, as illustrated in table 2.2, and by far the highest score for impact with 80 %, more than double of

MARIE and MissionLab who shared the second place. The popularity of Player/Stage is highly interesting, as the framework did not score top results in either the features or usability scores. What attracted users to the framework was that it was designed directly to be a programming interface, rather than a development environment or enforcing any types of architecture. This caused Player / Stage to be a good choice for researchers who wanted to get into robot development and research quickly, without the hassle of writing hardware device drivers and communication infrastructure. The popularity also produced a number of ready modules for sensor processing and simple differential motion control, but due to the programming interface ideology and lack of system architecture, no higher level modules were ever introduced into the Player / Stage distribution. The principles of Player / Stage were carried on by the key developer Brian Gerkey, who was requited by Willow Garage and brought into the foundations of ROS.

From the 9 frameworks reviewed in the paper by Kramer and Scheutz (2007), TeamBots stopped active development in 2000, Player / Stage in 2010, Pyro in 2007, CARMEN in 2008, MissionLab in 2006, ADE in 2004, Miro in 2009 and MARIE in 2009, leaving only ARIA to remain under active development, as it is commercially driven by Adept Mobile Robots to support their range of mobile robot research platforms. So where short-term development in research projects might have benefited by using one of these frameworks, long term achievements suffer when development stops.

A clear lesson learned from the pioneering efforts in robotic frameworks was that impact was not created alone by all research groups releasing their framework to open source. For example, the research in the community itself which formed around Player / Stage was what made the impact of the framework, not only the core team itself.

The halt in development for a large part of the pioneering frameworks has caused as loss of the developed functionality that was too closely linked to the framework. This phenomenon has changed the approach towards open source development for robotics slightly. Recent initiatives like OpenSLAM (Stachniss et al., 2010) and the Point Cloud Library[10] (PCL) were initiated to isolate and drive the development of functional libraries without linking them too closely to any robotics framework.

2.4.2 Hierarchical architecture structures

Early architectures like the classical NASA NASREM architecture (Albus et al., 1989) introduced the Sense-Plan-Act (SPA) principles in multiple layers, where each layer performs a closed loop of sensing, planning and acting, as well as producing refined information to a subsequent layer of higher level SPA control. The NASREM model is illustrated in figure 2.12.

[10]Point Cloud Library, http://pointclouds.org/, 2012

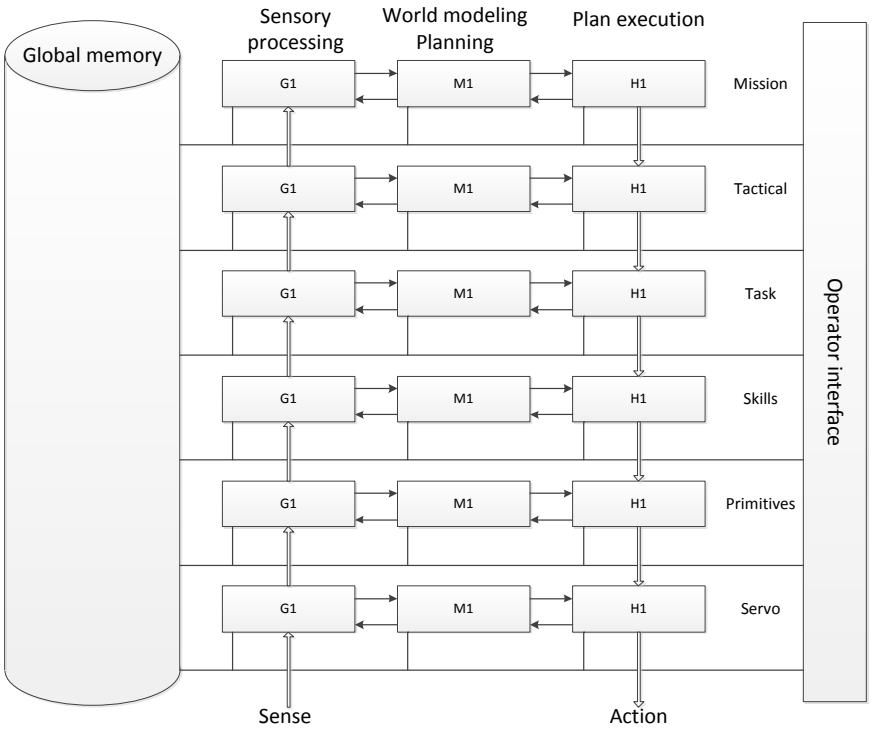

Figure 2.12: The NASA NASREM architecture model (Albus et al., 1989) introduced the SPA principle and has been a reference model for numerous robot architectures.

The SPA architectures were a predominate architecture for robotic systems throughout the 1990s, but like the original JDL Data Fusion model, the SPA suffered the disadvantage of being interpreted as a canonical implementation guide rather than a principle architecture. This problem became evident when open source robotic frameworks changed the development from being in one group with a common understanding within one department to a larger community. This change proved, that adopting new frameworks might be manageable if the software engineering approach is well crafted and enough impact is gained by the adoption, but to adopt a complex principal software architecture is much less appealing to new users.

The two unquestionably most popular robotic frameworks in history have been Player/Stage and ROS, which are interesting as these two frameworks explicitly are designed towards not enforcing any architecture. CARMEN is described by Montemerlo et al. (2003) to be an example of the 3T hybrid architecture but does not enforce it by any means. I will argue, that architecture neutrality is key success criteria for an open source robotic framework today, unfortunately, I also consider it a severe limitation to the application level achievements by the community of the framework. It quickly becomes anarchy between modules when system complexity increases and decision making must be integrated at multiple levels without clear guidelines of the roles of modules and interfaces between layers.

2.5 Modern Robotics Frameworks

In the following section I will look closer into two modern robotics frameworks: the Robot Operating System (ROS) and DTU MobotWare.

DTU MobotWare is the software framework developed at the department of Automation and Control at DTU Electrical Engineering and is the robotics frameworks used as development and experimental framework in the work presented in this thesis. To give a better understanding of the components and structure of the framework, as relatively through introduction to the framework is given in the following section 2.5.2 on page 28. For further details about the framework, please refer to (Beck et al., 2010).

The structure of ROS is investigated due to the popularity of the framework, and thus forms a baseline to ensure that any assumptions made throughout this thesis on communication architecture and component structure apply to the general case and not MobotWare specific details.

2.5.1 Robot Operating System: ROS

Fueled by the ambitious and mysterious Willow Garage, ROS has rapidly become the de-facto standard of robotic frameworks. ROS was initially announced as conceptually being an operating system for robots (Quigley et al., 2009). The motivation behind presenting ROS as an operating system was that traditional operating systems basically organized device drivers, software structures, process management, communication and graphical interfaces - so does ROS.

The fundamental ideas and concepts of ROS were created from experience and challenges in large service robot projects at both Stanford University and at Willow Garage. The fundamental goals of the framework were summarized by Quigley et al. (2009) as

- Peer-to-peer
- Tools-based
- Multi-lingual
- Thin
- Free and Open-Source

The fundamentals of a system using ROS are of a number of processes connected in a peer-to-peer topology. Connection peers could either be processes on the same physical machine or on the other side of a cabled or wireless LAN. Lookup to establish new peer-to-peer connections is managed in a central “master” service. This topology ensures independence between peers, easy connectivity and access to all data and functions from anywhere in the architecture. However, for large volume data traffic to or from many peers, the topology is quite inefficient.

As the only linkage between components or "nodes" in ROS is the peer-to-peer communications system, there is freedom to implement them in any programming language, as long as the communications library exists in the given language.

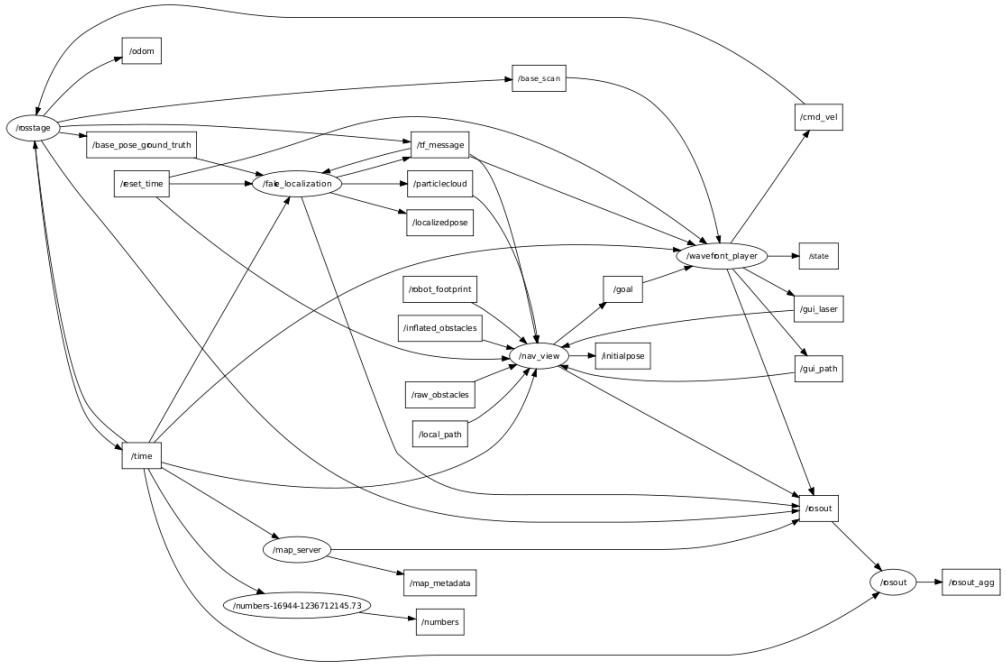

Figure 2.13: Visualization of ROS computation graph. Connections between ROS-nodes are created by request from nodes themselves in a peer-to-peer manner.

Similar to the Linux architecture, functionality in the ROS system is built into a set of tools, which are used to configure the system, as well as visualize peer-to-peer connections, debug nodes, and graphically plot sensor readings and other message data. Figure 2.13 shows an visualization of a ROS peer-to-peer topology instance for a mobile robot simulation using the Stage simulator (Vaughan, 2008).

As it is apparent in figure 2.13, the peer-to-peer topology can become fairly complex, but the tools in ROS make it possible to tap into any message stream to investigate the data to debug any module or combination of modules.

The mobile robot navigation in ROS is implemented in the *navigation stack*, who is a collection of nodes, which combined gives the basic navigation capabilities of a mobile robot base. The principles of the stack are fairly simple, as it connects to information from the odometry of a mobile robot platform and sensor streams, such as a laserscanner and outputs velocities to the motor controllers of the base platform. Using the navigation stack, the robot can be localized in a previously defined map and utilize path planning modules to avoid obstacles on the way to the goal. In general, this stack seamlessly combines a number of state-of-the-art modules and algorithms into one package for easy navigation

using mobile platforms.

Looking into the navigation stack from an application development perspective is, however, a bit disappointing. The control of the platform is performed by issuing a sequence of coordinates in C++ of where the robot should position itself like illustrated in listing 2.1.

```
MoveBaseClient ac("move_base", true);
goal.target_pose.pose.position.x = 1.0;
goal.target_pose.pose.orientation.w = 1.0;
ac.sendGoal(goal);
ac.waitForResult();
```

Listing 2.1: Assignment of the ROS navigation stack to move 1 m forward.

The use of the actionlib class `ac` in the example in listing 2.1 allows the user to preempt the execution of assigned goals or to monitor the execution while it runs. But as this example also illustrates, execution of larger tasks or development of mobile robot applications are simply implemented in C++ by sequencing raw move-pose commands. At the current state, ROS does not provide any scripting language to define higher level missions or to monitor task execution for error recovery.

Like Player one of the design philosophies of ROS is to be open and architecture neutral, which leaves the design of higher level architectures to each user group. As discussed earlier, this design decision has most likely played a part in the success of ROS as it seems much harder to share high-level architectures across communities than sharing low level signal processing modules. As a result, low level modules are no longer rewritten all over the world, but high level mission and application execution modules still are.

2.5.2 DTU Mobotware

DTU MobotWare is a mobile robot control framework developed at the Technical University of Denmark (Beck et al., 2010). The first modules date back to 1999 as reusable components to control mobile robots in research projects. Throughout more than a decade of research projects, master and ph.d. projects have formed the framework into a modular and plug-in driven software structure.

MobotWare has been born in a university research environment, which has been a central part to form its design philosophy: *Writing a software framework for autonomous mobile robots requires an effort that goes beyond one project.* This means that it is important to control development over a longer period. A university environment is a very dynamic environment, where students work on isolated short term projects and often face hard deadlines at the same time as they are finalizing their testing. Furthermore, most university projects are case or application driven, which often poses a need for making core system adjustments to suit the needs of the specific cases.

In such an environment it is difficult to enforce strict software development methods and rules. Often it means that the software written in a project is

abandoned when the student leaves after finishing his or her project, leading to rewriting the same functionality again and again, not only in different places in the world but even at the same department.

The MobotWare framework has three core modules:

Robot Hardware Daemon (RHD) Flexible hardware abstraction layer for real-time critical sensors

Mobile Robot Controller (MRC) Real-time Closed-loop controller of robot motion and mission execution

Automation Robot Servers (AURS) Advanced framework for processing of complex sensors and non-real-time mission planning and management

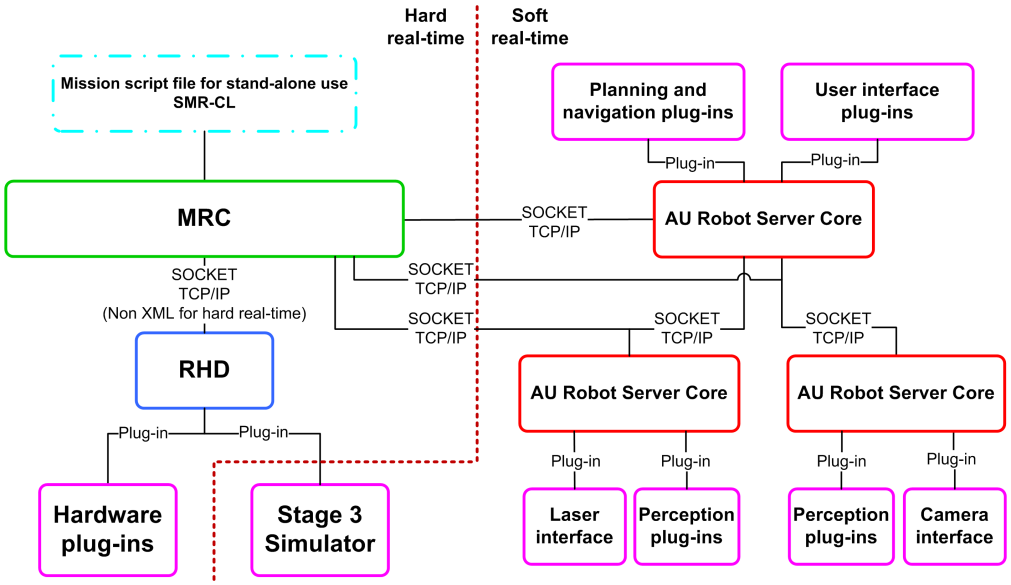

Figure 2.14: Overview of the MobotWare mobile robot control framework.

Using these three core modules, MobotWare implements a control architecture into the framework as illustrated in figure 2.14, which was deliberately avoided in many other popular frameworks like ROS (Quigley et al., 2009), Player (Gerkey et al., 2001) and OROCOS (Bruyninckx, 2001). RHD and MRC form the hard real-time control section and AURS takes care of all heavy and non-deterministic sensor processing and planning. The choice of having a layered control architecture has a number of benefits:

- There is a clear distinction between the hard real-time control section and the non-deterministic soft real-time section.
- Modules are easier reusable, as they are developed to play a specific role in a limited context.

- All data-reduction must be handled at the level where the data are locally available, rather than where a developer finds it most convenient at a given time.
- Developers tend to solve problems on the context where they currently are working. In an unstructured university development environment, this often causes modules to be a mixture of algorithms and decisions belonging on several levels. Architectures force developers to create general and reusable modules at the right levels to solve tasks.
- Stability is ensured by having the right code executed at the right level.

Core components are connected through low latency TCP/IP connections, that makes is possible to distribute the MobotWare components across multiple computer platforms, without the need of complex 3rd party communication libraries. This could be a low-power platform for real-time control and a high-performance platform for laser or vision processing, or it could be a development module conveniently running on the development machine, while all other modules are performing on the actual robot platform.

All communication protocols, except the RHD real-time sensor interface, are implemented in human-readable XML, which is efficient for debugging and testing, yet powerful for inter-process communications due to efficient and readily available parsers. By design decision, sensor data must be processed at the source and only reduced data are distributed, in opposition to e.g. ROS and Player. It makes the development doctrine slightly strict, but improves stability in a university development environment.

Development and research of new functionality within the framework is done through a plug-in interface on all system levels. This ensures that students and researchers have a well-documented entry-point into the framework that minimizes time spent on learning a complex system. Another strong advantage is that core stability of the MobotWare framework is always ensured, as failed or uncompleted projects can easily be removed - and successful projects can be saved!

The Robot Hardware Daemon (RHD), a real-time device server, has been developed to cope with the increasing diversity of the supported mobile robot platforms (Beck and Andersen, 2008). In opposition to related initiatives such as Player (Gerkey et al., 2001) or CARMEN (Montemerlo et al., 2003), RHD is limited to be a lightweight hard real-time hardware interface. Signal processing at RHD level is limited to protocol interaction and simple security functions as emergency stops.

RHD is a real-time synchronized variable database. The variable database structure provides great flexibility, but it also helps to enforce a clean cut interface between the various hardware formats and a user-manageable API without enforcing specific device abstractions. RHD is also the primary real-time control scheduler in the MobotWare framework, so much effort was used to analyze the

behavior of scheduling mechanisms in Linux, and to create a robust, lightweight and flexible real-time safe implementation.

RHD consists of a set of core components, and a range of specific hardware driver plug-ins. The core components include the variable database, TCP/IP server and the real-time scheduler. Plug-ins create the support of hardware devices, by managing hardware I/O, pre/post-processing and database interface.

Besides being a networked variable database, RHD is also the main real-time scheduler for low level robot control applications, such as MRC. The MobotWare framework is expanding to support new robots and as some of these hardware platforms are big, powerful, and heavy, fault tolerance and especially real-time performance are critical.

Hard real-time performance is not generically supported by the Linux kernel, as there is no support for kernel pre-emption. In projects, as RTAI (Mantegazza et al., 2000) and RT-Linux (Bruyninckx, 2002), this is solved by patching the kernel with a second scheduler, which allows tasks to work in kernel priority levels and with full kernel pre-emption. Our RT-implementation of choice is the Linux Real-Time Application Interface (RTAI). Linux itself provides some means of achieving soft real-time performance. RHD is capable of operating satisfactory on a standard Linux distribution and with enhanced stability and real-time performance using the RTAI scheduler. For hard real-time performance RHD uses the LXRT module and RT-FIFOs from RTAI to obtain real-time scheduling in user space.

The Mobile Robot Controller (MRC) is providing the basic low level real-time control of the mobile robot platform. It uses RHD (Robot Hardware Demon) as an interface to the actual robot hardware. MRC has the following features

- Odometry
- Motion controller
- SMR-CL interpreter (Andersen and Ravn, 2004)
- Socket interface to high level controllers
- XML-based socket interface to sensor servers
- Socket interface to RHD
- XML-based configuration file
- Calibration support for odometry, line-sensors and distance sensors.

The odometry is based on wheel measurements and inertial sensors, e.g. gyros. It is configurable using an XML-based configuration file for several kinematics: differential drive, Ackerman steering and vehicles described with linear and angular velocity (v, ω). The motion controller provides the basic path control based on odometry as well as sensor based movements for the following SMR-CL commands:

fwd	Point to point forward motion.
turn	Turn around center.
drive	Continuous linear motion.
turnr	Circular movement with given turning radius.
stop	Stop the robot and idle motor controllers.
followline	Follow a line on the floor (painted or buried wire).
followwall	Follow a wall using e.g. infrared distance sensors or laser scanner.

Small Mobile Robot Control Language, SMR-CL (Andersen and Ravn, 2004), (Jørgensen et al., 2008) is an interpreted language intended for control of mobile robots. The language supports sequences of basic robot actions with multiple criterions for seamless motion gluing as well as standard mathematical expressions. The system has two kinds of variables, system variables that reflect the state of the vehicle and user variables that are created the first time they are used in an assignment clause. The language is intended to be used in the tactical layer just over the control layer, i.e. it is fast enough to shift between control strategies and react to state changes in real time. SMR-CL provides the bridge between the hard real-time demands of the control layer and the soft real-time demands of the planning (strategic) layer. The language may be used in two ways, either as scripts runs directly from text files or as a command language through a socket interface.

Automation Robot Servers (AURS). Sensors with high data volume - like cameras and laser scanners - usually require time consuming data processing. This is not easily compatible with the real time requirements of the motion control. Each sensor is often used for more than one purpose, e.g. the laser scanner may be used for detecting obstacles, for wall following, for human detection and emergency stop. An architecture, which supports sharing of sensor data, is therefore essential.

More than one processor unit is often needed when processing data from these high volume sensors. A standardized communication interface is therefore beneficial. In a university environment new functionality is often developed by students. It is therefore important, that the development can be done in independent groups and within a limited time frame. The architecture and programming interface must therefore be reasonably simple, and it must be easy to add (and remove) the developed modules without influencing other parts of the software.

The MobotWare solution to these challenges is a set of servers called AURS. A typical AURS configuration consists of a camera server, a laser scanner server and a mission management server. Each server consists of a server core that provides a number of general services to the functional plug-ins. These services include:

- Management for loading, unloading and configuring plug-ins dynamically
- Communication services (socket based)

- Event handling (generation and implementation)
- Server wide shared plug-in variable tree
- Plug-in to plug-in communication
- Direct access to base plug-ins (like laser sensor plug-in and pose history)
- API for common tools (like linear algebra, coordinate conversion and image algorithms from openCV.
- Data logging and replay services

As an example the perception laser scanner server could be configured as shown in figure 2.15. There is a plug-in to maintain the recent robot pose in each of the maintained coordinate systems (odometry and map coordinates), these are capable of providing the robot pose at any given (sensor) time. The laser scanner plug-in provides (raw) laser scanner range data and the laser scanner pose - relative to the robot. The other plug-ins use the sensor data to create and maintain a specific perception model.

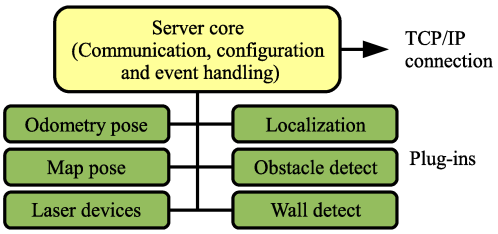

Figure 2.15: The AURS structure, with server core and functional plug-ins.

The communication to and from a plug-in (apart from the internal plug-in to plug-in function calls) is handled by the server core using XML based messages. Each plug-in handle commands formatted into one or more XML tag types, and thus can share the same server provided socket connection.

A plug-in can also trigger an event - an event could be that a new laserscan is available, the server may then be configured to react by sending an XML command to other plug-ins, e.g. to update the wall and obstacle model. Update of the obstacle model could trigger a further event, e.g. to start a new behavior generation. As described here, both sensor processing and part of the perception model are handled in one server. This is intentional, as the high volume data flow from the sensor plug-in to the perception plug-in in this way is kept internally in the server. The reduced data flow from the generated model is communicated to other servers only.

The server structure allows both client-pull data flow - send a command and get a reply - and server-push data flow, where data are sent as a response to an event. Timed events are supported directly by the server core. A large number of plug-ins are available, these include:

- a laser scanner plug-in, supporting Sick and Hokuyo scanners

- a camera device plug-in
- a MRC interface plug-in for easy access to real time data and behavior commands
- a road detection plug-in using a slightly tilted laser scanner (Andersen et al., 2006b)
- an obstacle detection plug-in using laser scanner data
- an obstacle detection plug-in using stereo camera data
- a road detection plug-in using single camera data (Andersen et al., 2006a)
- a localization plug-in combining laser scanner and a-priori map (Tjell and Hansen, 2008)
- a plug-in that implements a rule based mission scheduler
- a mission manager plug-in (Beck et al., 2009)
- obstacle avoidance plug-in based on visibility graph.

Data recorded from laser scanner, camera, GPS and robot pose[11] can be replayed based on log files from each of the relevant plug-ins and synchronized by the server core. The replay ability eases development of especially perception plug-ins significantly.

An operator interface is available to interact with all servers and all plug-ins, and displays also some of the more complex data structures. An example from the operator display is shown in 2.16.

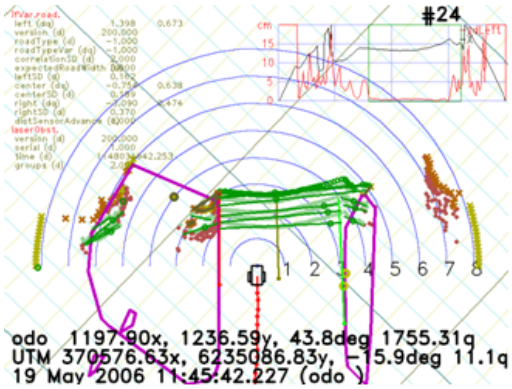

Figure 2.16: Operator interface example, illustrating traversable road (green) and obstacles (purple) after treatment of the perception plug-ins.

[11] Pose: position and orientation

2.6 Task Execution and challenges

The obvious challenges when considering mobile robotics are about planning motions, perceiving the environment, navigating robots and avoiding obstacles. These challenges are hard, but have been tackled by research to a level where a substantial number of algorithms and methods exist to present a solution to each challenge.

So, what is left? The problem with operating in natural environments is that a myriad of things can go wrong. Under these circumstances, even the most well proven methods can solve a challenge in 98 % of all cases, or maybe 99.9 %, yet there is still the last fraction of situations where the methods prove to be inadequate. Most of these times, the robot system has the knowledge of the error happening, but no mechanisms to handle what to do, except just continuing along a mission following a set of basic assumptions which no longer hold.

Having just one strategy to solve a problem is unnatural to us humans. Most problems are solved iteratively, by having a prototype strategy for solving the problem, but if methods fail or new information is learned, we revise our strategy and switch method. This could also be the case for robot systems, but generally is not.

Work not covered in this thesis, but presented in (Beck, 2009; Beck et al., 2009) investigates the use of mission management to employ multiple strategies for executing navigation missions. The proposed system is capable of switching between strategies in hard real-time by constantly feeding forward navigation alternatives across the hard/soft real-time boundary in MobotWare. Hereby the real-time motion controller MRC was capable of switching to secondary navigation options if the basic assumptions for the current navigation failed. Combination of this general framework for real-time behavior switching with a module for automatic mission planning / re-planning presented a reasonable solution to the “no plan B” problem within MobotWare. The proposed mission management architecture is illustrated in figure 2.17.

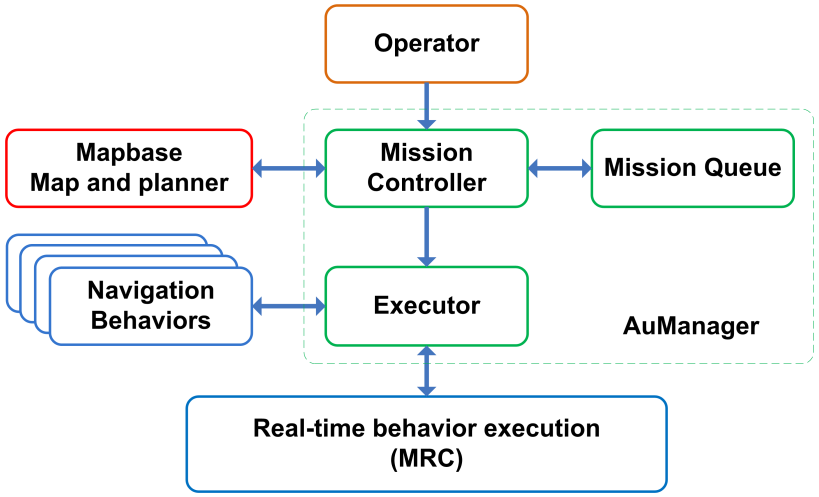

Figure 2.17: Architecture of the auManager mission management plug-in for AURS. Navigation behaviors are processed and pre-loaded into MRC, which allows switch of behavior in real-time if errors in current navigation strategy are detected.

Alternative solutions can be selected in case of errors if the proper execution architecture is in place. But sometimes failures of navigation algorithms are not detected within the robot systems at all. The environment might have changed so significantly that the localization system mistakes the current location with an entirely different in the map, or an event is missed, which causes the robot controller to be in an entirely state in the robot mission code than the robot is in reality.

This loss of synchronization between reality and the virtual representation within the robot controller is a significant problem as the robot can continue into hazardous situations without having any knowledge of the problem.

Combining the internal belief of the robot control system with measurements of the environment is one of the key challenges addressed in this work on situation assessment. I address the challenge of creating meaning from relations between observations, both internal and external, without having any knowledge about the system models, program or intentions. Observing these relations from a spatio-temporal perspective to form an understanding of the current situation and possibly even predict future situations is one proposed solution to improving the capabilities and reliability of mobile robot applications.

2.7 Summary

This chapter presented the current state-of-the-art within mobile robotic applications. Mobile robots in commercial applications within industry and domestic settings are employed in solving routine functions such as transporting large volumes of goods around warehouses or vacuuming the floors of our homes. Application innovation has created new epoch-making solutions using mobile robots, such as the Kiva solution, without any significant technical breakthroughs but simply by using simple robot platforms in genius application settings.

The state-of-the-art research applications push the limits of sensor systems and environment perception, like the Google Self-driving car, which employs 3D environment reconstruction using a 64-beam 3D laserscanner or the Claas Agrocom vision sensor who tracks rows and crop edges in real-time to control tractors autonomously. Confronting the limits of technology also increases the likelihood for errors, either in algorithms or by the perception capabilities of the sensor systems.

An in-depth study of a number of common open-source robotic frameworks investigates their capabilities of handling these errors. Either by explicit modules for error recovery or by advanced mission control to adapt or re-plan missions to possible events of errors. Very little of this capability exists in the frameworks investigated.

Robot application might fail due to errors in critical algorithms or because the synchronization between the vitual representation in the control system and the real environment is lost, causing the robot to continue its mission in a completely wrong scenario. Solving the challenge of detecting these errors by assessing the situation of the robot is the goal of the work presented.

3

Situations for Mobile Robots

Situation:
The combination of
circumstances at a given moment.

The great British dictionary

In the previous chapter I presented state-of-the-art within mobile robot applications and showed how the current work of the mobile robotics research community has made significant contributions within the field of navigation, perception and control of the mobile robots. However, support for higher level control and especially for reliability enhancement has not found its way to play a significant role in the community efforts for mobile robotics frameworks.

In mobile robot applications, any number of things can fail as the robot operates in dynamic environments which could change critically or sensors could miss important measurements. Commonly known failures might be handled within motion skills themselves. However, many problems are not evident by only considering the measurements, and some situations will only be considered problematic in certain critical contexts, but considered normal in other contexts. Assessing situations for the mobile robots will produce this knowledge and provide crucial information for the robot controller to deal with errors and critical situations.

In this chapter I will introduce a formal definition of situations, and how situations can be represented in terms of primitives. A number of circumstances can form the basis for a situation. In order to accurately describe a situation, also the spatio-temporal properties of these circumstances are required to describe the situation state, as well as the relationship between the circumstances is required to describe the situation dynamics.

The specific case of critical situations for automated systems is discussed. A necessary criterion for successful planned behavior is that the basic assumptions upon which the application have been designed, hold throughout the execution of the application. Situations where these assumptions are endangered or fail can be identified as critical and are particularly interesting for situation assessment.

In the last part of the chapter, I introduce three types of strategies for situa-

tion assessment. The strategies are based on the assumption that situations for the mobile robots can either be considered normal or anomalies when they are compared to the goals which have been programmed for the robot application.

3.1 The importance of situations in automation systems

Handling errors in automated systems is generally very difficult. In manually operated systems, the operator has complete situation awareness of the system state, the process and how to control it. When systems become semi or fully automated, the operator changes role to being a passive observer instead of actually being a part of the process as well as the type of feedback given to the operator changes. There are numerous examples of operators who are reportedly unaware of automation system failures and does not detect critical system changes when acting as monitors for automated systems (Endsley, 2006).

For humans this problem is known as the "out-of-the-loop" problem. Humans are notoriously poor observers if little or no interaction is required. And when errors are detected, humans need additional time to investigate the state of a system in order to sufficiently understand the problem situation. Time spent in these steps can be critical for the system, with results ranging from decreased performance to catastrophic failures with major consequences.

For robot systems, human operators can be a crucial component to resolve complicated error situations, as the problem solving capability of humans still surpasses what can be accomplished using AI by several magnitudes. But in order to increase the effectiveness of humans as error resolution agents, they need to have presented clear detections of errors and comprehensive descriptions of the error situations in order to contribute efficiently to the error resolution.

Having detailed knowledge of the current situation can help more than the human operator, as automation systems can be designed to utilize the situational knowledge in order to react to certain situations or anticipate future events.

3.2 Situations

A focal point throughout this thesis is the term *situation*. Before I connect further analysis to this term, it is necessary to address and identify what a situation is in a robot context. The discussion of situations has played an important role within many branches of science.

Defined by the great British Dictionary, a situation is *"the combination of circumstances at a given moment"*. Wikipedia defines a situation as *"an abstract object relating to a position (location) or a set of circumstances* [12] *"*.

From these situation definitions two key elements can be extracted as the core contents of a situation definition, thus I propose a situation to be described at the abstract level by

[12] Wikipedia, URL: http://en.wikipedia.org/wiki/Situation, June 2012

- A combination of circumstances
- A relation between the circumstances.

In the early 1980s, Barwise and Perry (1983, 1981) made a large contribution on a framework for describing and reasoning about common sense and real-world situations, which they named Situation Semantics. Their focus was especially within the domains of theoretical linguistics, philosophy and applied natural language processing. In *Situations and Attitudes* they state:

> *Situations is basic and ubiquitous. We are always in some situation or other. Human cognitivity categorizes these situations in terms of objects having attributes and standing in relations to one another at locations - connected regions of space-time (Barwise and Perry, 1981).*

Formalizing this, they define situations to be formed by the primitives: objects, relations and locations. From this set of primitives, Barwise and Perry (1981) identify a formal definition of situation components, which expand the earlier proposed definition of situations

- a set A of individuals $a_1, a_2, ...a_n$;
- a set L of space-time locations $l, l_1 ... l_n$
- a set R of relations $R = R_0 \cup R_1 \cup ... R_n$, where R_n consists of the n-ary relations of individuals and locations.

Barwise and Perry (1983) advocate an ontology for situations, stating that a situation is roughly a set of n-tuples, where each n-tuple contains the situational elements: objects, a relation, and a spatio-temporal location at which the objects have this relation.

This formulation of a situational description has been the widely accepted, but the remaining work of Barwise and Perry (1981, 1983) has received a lot of critical review throughout the 1980's and 1990's, as the primary contribution of their work was focused on deriving the grammar and semantics where natural language could be fitted. Natural language and philosophic analysis has a very wide scope and is traditionally not bound to formal rules and grammars, which makes it difficult to propose a universal and agreeable model.

The world of robotics can be observed as less open and slightly easier to formalize, based on two assumptions:

1. Robot systems are based on controls who only work on well-defined and known parameters.
2. The theory will be beneficial in the world of robotics if it can contribute to problem solving on real-world situations, even if the proposed theory does not cover and handle all context specific and hypothetical situations.

In summary, there is a general agreement in literature that a situation can be described as a collection of a number of circumstances (objects) and their relation. But as proposed by Barwise and Perry (1981) the term *circumstance* and the term *relation* implies that not only is a situation defined by a number of object parameters, but the spatio-temporal location is just as important.

The aim for this thesis is to create a framework to recognize and react to situations that can be generalized for robots. Butterfield (1986) criticizes Barwise and Perry (1983), by stating that their situational semantics only *act like partial models or parts of possible worlds (Butterfield, 1986).* Being a strong critical argument to Barwise and Perry (1983), it perfectly supports the hypothesis of this thesis. The goal is to have a model who acts and works as a partial model of a possible world, as it will allow us to work with incomplete or generalized situations. Robots can only perceive a limited part of the world based on its sensors and internal world model, and thus the situations it can recognize are only partial impressions of the world.

3.2.1 Critical situations

A particular interesting instance of a situation is the so called critical situations or crisis. As stated by Barwise and Perry (1983) we are always in one situation or another, but for reasoning and decision making the particular interesting situations are the critical situations. A critical situation has been defined in literature as:

> *An unstable or crucial time or state of affairs in which a decisive change is impending; especially: one with the distinct possibility of a highly undesirable outcome* [13].

> *A disruption that physically affects a system as a whole and threatens its basic assumptions, its subjective sense of self, its existential core.* (Pauchant and Mitroff, 1992)

By definition, a critical situation is a situation, which has critical impact in the world model it describes. Known from laboratory testing of research robot systems, most errors happen because the robot operates on assumptions which no longer holds. It could be that a localizer has lost the robot position, a sensor misreading or hardware malfunction. In all cases, the robot system will continue to operate on assumptions which are false and require some decision making to avoid malfunction.

3.2.2 Example of handling critical situations

When doing system integration of automation systems, it is common practice to analyze the system process for critical situations. Figure 3.1 illustrates a timeline for an automation system process. Within such a process, there are non-critical phases where little can go wrong, such as process items being transported

[13] The Merriam-Webster dictionary

on a conveyor or being moved by a robot. But in certain phases, often in connection with transitions or complex handling, there are critical situations i.e. where the robot could miss a grip or items could get stuck. In figure 3.1 the critical elements in the time line are marked with red circles.

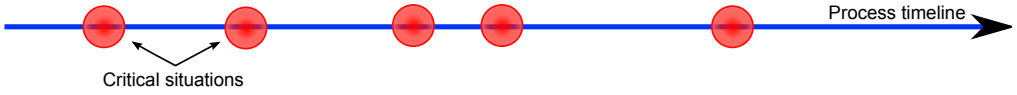

Figure 3.1: Critical situations within a process in an automation system. Most of the time processes are known to be reliable, but at certain moments, identified critical situations must be handled.

The traditional automation systems approach to handling critical situations is illustrated in figure 3.2. When a critical situation is identified, the integration engineer dives into the problem and writes a piece of exception handling code. After a lot of test iterations, the exception handling code has become quite elaborate and handles the unique critical situation. If a number of critical elements were found within the critical situation, it is handled through a similar iterative process.

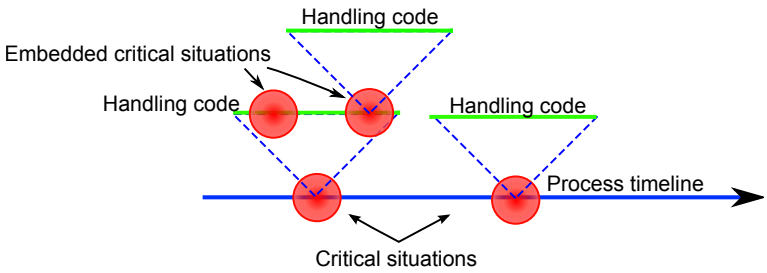

Figure 3.2: Handling critical situations in automation systems are traditionally done by identifying the situations and creating customized handling code to robustify the situations. In an iterative testing process it is identified if the critical situations might contain additional embedded critical situations and they are handled in similar manner.

Although the method for handling critical situations by analyzing them and optimizing/removing the critical elements is an intuitively sound method of improving robustness of an automation system, the key drawback is that the method only solves the single unique critical situation in a completely unreusable manner. If there is a change in products the system handles or mechanical wear on the system, the assumptions which formed the solutions in figure 3.2 no longer hold. For mobile robots, even in semi structured environments, it cannot be assumed that very detailed assumptions hold, even in normal operation.

3.3 Strategies for Situation Assessment

Situations exist as certain sequences of spatio-temporal relationships. The chains of events are what defines a situation, just as much as the events themselves.

For any type of robots, behavior is programmed to follow certain behaviors and often using various types of feedback to ensure that the system truly follows the programmed behavior to the best of its ability. As a generalization of situations for robots they can fall into two categories. Situations can either be considered to be normal, which follows expected spatio-temporal patterns, or as anomaly where the situation deviates significantly from the expected patterns.

Anomalies can occur when any of the basic assumptions for an application fails, e.g. if motors fail, environment changes unexpectedly or if crucial algorithms fails. Anomalous situations can happen at any time when unexpected events force the robots to deviate from its planned behavior, but in the situations defined as critical situations there is an increased likelihood of failures happening. The critical situations can be identified as described earlier in section 3.2.2 on page 42 and monitored specifically.

In total, four strategies for assessing situations from spatio-temporal patterns are identified

Type 1: Detection of known spatio-temporal relations.

Type 2: Deviation from known spatio-temporal patterns.

Type 3: Detection of known critical situations.

Type 4: Unpredictable "Black Swan events".

The following sections define the characteristics of the situation assessment strategies.

3.3.1 Type 1: Detection of known spatio-temporal relations

Although mobile robots often operate in environments which are fairly dynamic, much of its behavior is predictable and in long term operation it will become repetitive. Any number of parameters from within the robot controller and from measurements of its environment can be assembled and their relations can be investigated in time and space. A result of this will be patterns which describe the situations the robot passes through. Depending on the choice of parameters, some of these patterns will make sense, others not.

A natural example could be to investigate the pose (x, y, θ) of a mobile robot while it performs a repetitive sequence of motions. Spatio-temporal relations between these parameters will describe the characteristics of the motions, e.g. at which position and orientation it had at a certain time, and what motions that followed. With such a model of the known robot behavior, it is possible to recognize situations for the robot, when it follows the sequence of motions again and it is also possible to predict what might happen in the near future for the robot.

A critical scenario for mobile robots is precision navigation operations. These situations can be docking, positioning for manipulation, or other motion tasks where close proximity to objects is required. It is particularly critical as there is

an reasonable amount of absolute inaccuracy associated the mobile robot localization methods and localization often are performed in relation to a global or local coordinate frame, not in direct relation to the objects in close proximity.

An example of precision navigation is the CoffeeBot demonstration at Automatica 2010 at the DTI stand, illustrated in figure 3.3.

Figure 3.3: The Coffeebot mobile platform was used for fetching coffee to visitors at the Automatica 2010 at the DTI stand. Precision positioning of the robot was essential to ensure that the cup holder caught a cup and coffee was poured into the cup.

The CoffeeBot followed a detailed sequence of motions in order to position itself to catch a cup from a cup dispenser and afterwards move to have coffee brewed into the cup from the vending machine. In both situations, it was critical to have an exact position as otherwise coffee would be spilled or the robot would crash into the machine. The movement pattern of the demonstration is illustrated in figure 3.4.

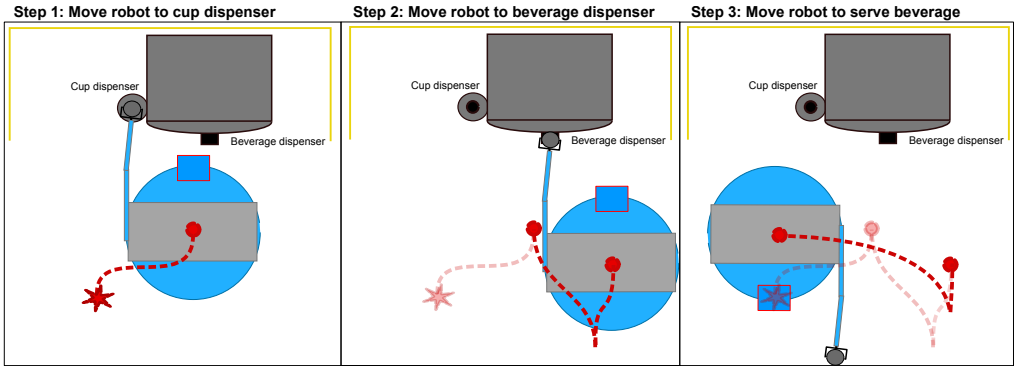

Figure 3.4: The coffee fetching behavior follows three steps. In step 1, the robot moves to the cup dispensing unit to have a cup dropped into its cup holder. In step 2, the robot moves backwards and then forward to position the cup holder underneath the coffee brewer. Finally in step 3, the robot moves (gently) to serve the coffee.

Precision was crucial for the application, so the robot control was designed to follow the exact same trajectory at each run. Using a spatio-temporal model of this sequence it is always possible to derive the current situation of the robot based on its current state in the model and its future behavior can be predicted from the model, without having any knowledge of how the mission control is designed.

A result of using this situation assessment strategy, models can be designed to capture the characteristics of certain elements within a mobile robot mission. From prior knowledge of the robot behavior, the current situation of the robot can be assessed and its future status can possibly be predicted.

Situation assessment using this strategy is specifically investigated in the AGV Situation characterization experiment in section 7.4 on page 132.

3.3.2 Type 2: Deviation from known spatio-temporal patterns

When spatio-temporal patterns are known parts or complete sequences of mobile robot applications, they describe with some certainty how the application is designed to be executed. Any deviation from these expected spatio-temporal sequences represents a possible critical situation. Figure 3.5 illustrates a few situations using agricultural machinery, where the current situations in environments clearly deviate from the predicted models.

Figure 3.5: Examples of using agricultural machinery, where operator situational assessment was inadequate to assess the change in environment and thus crashed.

Returning to the CoffeeBot scenario, the robot trajectory was defined in very clear detail. A localization system was used to continuously update the robot positions in relation to the wooden frame behind the coffee machine. This worked fairly well, but at certain instances, the localizer did not capture the lines of the wooden frame correctly at the first initialization, which caused a bad localization of the robot. This critical situation was evident already from the beginning of the mission and could be detected by a poor localization quality estimate as

well as unusual distance measurements to surroundings by the laserscanner. If these signatures were captured in a spatio-temporal model, a prone failure of the CoffeeBot application could be detected in good time before any actual failure and spilled coffee.

Using this situation assessment strategy, deviations from type 1 models can be used to predict the emergence of possible critical situation. These situations could either represent the likelihood of future application failure, or they could represent any predicted change in the application, which also could benefit from being handled from a situation assessment perspective.

Using this situation assessment strategy is investigated in the detection of errors in AGV localization tracking experiment in section 7.3 on page 128.

3.3.3 Type 3: Detection of known critical situations

In some military situation assessment contexts, it has been claimed that there are too many patterns of normal behavior to use these for general situation assessment solutions (Rogova, 2011). For mobile robot systems, a large part of the behavior of the systems is following static programs and even using feedback control algorithms to follow certain patterns. I argue that in a large part of these applications, commonly reoccurring patterns can be extracted and used successfully for situation assessment.

Rogova (2011) argues that in contrast to normal patterns, there are less abnormal patterns, which lead to this third strategy for situation assessment. Instead of creating spatio-temporal models of normal behavior for the robot, a model of known critical situations or abnormal behavior can be created. By knowing the signatures of critical situations, they can be detected early when the robot changes into this behavior and thus the possible error can be avoided instead of being recovered from. Enabling robot controllers to do error avoidance instead of error recovery will be a significant step forward in development of reliable mobile robot systems.

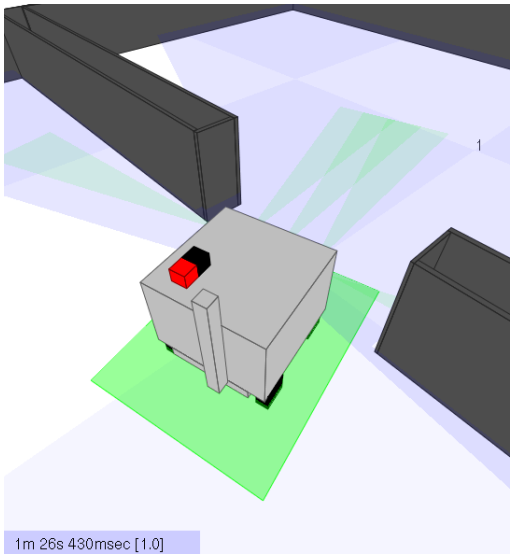

Figure 3.6: A simulated robot moving through a narrow passage, where orientation must be controlled carefully throughout the passage to avoid collision.

An example of a known critical situation is moving through a narrow passage, such as doorways. The robot must be properly centered in the doorway and control its orientation through the entire passage to avoid collision. Figure 3.6 illustrates a simulated robot moving though a narrow opening, which is investigated in the Narrow passage experiment in section 7.1 on page 110. The purpose of the experiment is to create a general narrow passage detector to assess the critical situation and predict future narrow passages for the robot.

3.3.4 Type 4: Black Swan events

Black Swan events was introduced by Taleb (2001, 2008) as events which are defined as

- The event is a surprise to the observer.
- The event has major impact.
- After the first recorded instance of the event, it is rationalized by hindsight, as if it could have been expected.

Figure 3.7: Black Swan events are surprising events with major impact.

This type of events are unavoidable in robotics and virtually all experimental practitioners within robotics have experienced their brilliantly crafted experiments fail due to unsuspected circumstances which they probably could have anticipated, but did not.

Solution to this type of situations is not explicitly covered within the work of this thesis. The type 2 detection of deviation from known patterns strategy covers detection of unexpected behavior within well-known situational patterns. However, this strategy only considers deviations which are represented by the parameters selected to describe the situation. Many Black Swan events can be imagined which go completely undetected by the sensors of the robot until the sudden event of failure.

3.4 Summary

This chapter introduces the concept of situations as a combination of circumstances and the relations among these circumstances. This definition is expanded by Barwise and Perry (1981) as a formation of three primitives: a set of objects (circumstances), a set of spatio-temporal locations and a set of relations.

From this definition the special case of critical situations are discussed. Critical situations or crisis are situations where the likelihood of failure is unacceptably high, and thus must be handled. For mobile robots, critical situations are most likely to occur when the basic assumptions for the robot applications fail. Such situations are likely to cause failure, despite of the application control working as intended, as failed assumption could be a hardware fault or a missed sensor reading.

Finally four strategies to situation assessment in a mobile robot context are presented, where three are explicitly addressed in the context of this thesis: Detection of known spatio-temporal relations, detection of deviation from known spatio-temporal patterns and detection of known critical situations. Dealing with unexpected “Black Swan” events are not addressed in this work, except when the events fall into the strategy of detection of deviation from known patterns.

4

Situation Assessment

> We are drowning in information but starved for knowledge. The current level of information is clearly impossible to be handled by present means. Uncontrolled and unorganized information is no longer a resource in an information society, instead it becomes the enemy.
>
> *The International Society of Information Fusion.*

Situations play an important role for us humans to interpret what is going on. And when we combine our assessment with experience, we find that resolving a proper solution is quick and efficient, even without us knowing all details.

For humans it is natural to use templates of situations to match our current situation and achieve situation awareness. This enables us to quickly comprehend the current status and predict what will happen next. This chapter introduces situation assessment as the process used to achieve situation awareness.

In order to transfer this ability to the domain of mobile robots, I investigate how situation assessment has been approached in science. Within the domain of information fusion, situation awareness has played a role since the late 80s and in particular from a military perspective. In information fusion, situation assessment is considered a higher level fusion process, which fuses results from lower level fusion processes and other available knowledge to form an on-line assessment of the current situation.

Another field which has addressed situation awareness and situation assessment is the research within human factors. The field of human factors research has the focus on the achieved situation awareness of human decision makers in complex dynamic scenarios, such as air traffic control and control of large scale automation systems. In this context the connection between human and computer systems is crucial and much work has been focused towards designing systems to automate or aid situation assessment.

Analysis of the presented work leads to the final contribution of this chapter, which is a proposed modeling framework for situation assessment. The proposed model is using the spatio-temporal Extensible Markov Model framework by Dunham et al. (2004) to model the spatio-temporal characteristics of situations. Furthermore, a situation prediction engine is proposed to perform continuous maximum likelihood calculations in the EMM to predict future events.

4.1 Information Fusion

Sensor fusion and state estimation are core elements within research and teaching in control theory and signal processing. Applications for sensor fusion are found within state estimation, dynamic system modeling and system identification and often are related to process control. However, there are a large number of other purposes where multiple and possibly heterogeneous information sources can be merged into a higher level of knowledge. This field is known as Information Fusion.

Information fusion is the synergistic integration of information from different sources about the behavior of a particular system, to support decisions and actions relating to the system. So in contrast to traditional sensor fusion, information fusion is targeted towards determining the behavior of systems, not only to control the dynamics of the system but also to support decisions and actions associated with the system. Information fusion includes theory, techniques and tools for exploiting the synergy in the information acquired from multiple sources, for example sensors observing system behavior, databases storing knowledge about previous behavior, simulations predicting future behavior and information gathered by humans.

Interestingly, the field of information fusion has attracted much attention from military research. The interest have been so explicit, that even textbooks distinguish between *Department of Defense* (DoD) and non-DoD applications (Hall and Llinas, 1997). Naturally, a key challenge for military intelligence is to understand what is going on in a very complex scenario of information. Military interests in information fusion range from robust real-time target tracking e.g. using moving radars, to multi target tracking and surveillance using sensors on ships, aircrafts, submarines, as well as ground- and ocean-based sensors and to form ”battlefield overviews”. Extracting useful information of these enormously distributed sensor systems is a complex fusion of information of different level of reliability and without specific dynamic system models. On top of the object identification and tracking tasks in battlefield overviews, DoD users seek to achieve higher levels of information about enemy situation (e.g. relationships between enemy entities, their relationships with the environment and higher level enemy organizations). Examples of sensor-based DoD applications are listed in table 4.1.

Specific application	Inferences sought	Primary observable data	Surveillance volume	Sensor platform
Ocean Surveillance	Detection, tracking, identification of targets/events	EM Signal, Acoustic signal, Derived observations (wake)	Hundreds of nautical miles	Ships, Aircraft, Submarines, Ground-based, Ocean-based
Air-to-air and Surface-to-air defense	Detection, tracking and identification of aircraft	EM Radiation	Hundreds of kilometers	Ground based, Aircraft, Ships
Battlefield intelligence	Identification of potential ground targets	EM Radiation	Tens to hundreds of kilometers	Ground based, Aircraft
Strategic warning and defense	Detection of indications of impending strategic actions	EM Radiation, Nuclear related	Global	Satellites, Aircraft, ground-based

Table 4.1: Examples of sensor-based DoD applications of information fusion, from (Hall and Llinas, 1997)

Besides the sensor-based approaches from table 4.1, military intelligence research contributes significantly to the information fusion community within the field of soft information fusion and higher level fusion. Soft information fusion often deals with elements such as natural language and other soft data types, which can involve a lot of ambiguity. Examples of soft fusion are Prentice et al. (2010) who presents a framework to apply a graph-based technique for fusion of natural language messages or Gross et al. (2010) who uses ”dirty graphs” to do template graph matching on uncertain soft information. Higher level fusion involves the complex and contextual information that is subjectively reasoned and analyzed to form higher level knowledge.

A second broad community which addresses data fusion problems is the academic/industrial/commercial community, which has a much wider scope of applications. The key issue is to extract enriched knowledge from a large body of information, either gathered from a distributed sensor system, or by processing information from high volume data sources, such as network traffic analyses, telephone statistics, web services or Twitter. Shroff et al. (2011) presents a blackboard architecture with a locality sensitive hashing to prune and analyze data from the assumption that the information sources are locally sensitive. The blackboard system was tested using Twitter feeds to ensure supply chain reliability. Twitter feeds were explored for topics like natural disasters or disturbance nearby important production facilities in a company supply chain. Another important application area is cyber security. Schwoegler et al. (2011) from Raytheon Company applies Multi hypothesis tracking, normally known from estimation in (kinematic)dynamic systems, to follow the dynamics of cyber intrusion sensors and automate the identification of individual attacks.

The pirate situation in the Gulf of Aden close to Somalia has attracted particular interest in the Information Fusion community, as a large part of the community is either defense affiliated, defense funded or defense subcontractors. Furthermore, maritime surveillance is an obvious application for higher level in-

formation fusion. Modern Automatic Identification Systems (AIS) makes it easy to get maritime vessel signatures and track their journey using existing radar installations. For example, a knowledge-based system, including a proposed representation of knowledge, inference engine, and series of rules is presented by Roy (2009); Roy and Davenport (2009). Unsupervised learning techniques using Gaussian mixture models to learn patterns of motion behavior are presented in (Garagic et al., 2009). Adaptive kernel density estimation is used in (Ristic et al., 2008) to model normal ship tracks and where departures from this model are considered anomalous. Lane et al. (2010) outlines five anomalous ship behaviors: deviation from standard routes, unexpected AIS activity, unexpected port arrival, close approach, and zone entry. For each behavior, a process is described for determining the probability that it is anomalous. Individual probabilities are combined using a Bayesian network to calculate the overall probability that a specific threat is present.

4.1.1 The JDL Model

A historic barrier to establish Information Fusion as an individual field of research and to facilitate technology transfer between application domains has been the lack of a unifying terminology. Information Fusion spans an enormous field of applications, and even within related military applications with similar approaches, different definitions for fundamental terms have been used, such as correlation and data fusion. In order to improve collaboration and communication between military researchers, civilian developers and system developers the (US Army) Joint Directors of Laboratories (JDL) working group for Data Fusion was established in 1986. The result of the effort was a process model for data fusion, formally known as the JDL Data Fusion Model, illustrated in figure 4.1, and a Data Fusion Lexicon (White, 1987).

In the Data Fusion Lexicon, the JDL working group defined data fusion as

> *a process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, and complete and timely assessments of situations and threats, and their significance. The process is characterized by continuous refinements of its estimates and assessments, and the evaluation of the need for additional sources, or modification of the process itself, to achieve improved results (White, 1987).*

Already from the definition of data fusion, it is clear that fusion needs to be done in a number of abstraction layers. There is a significant difference between estimates of position and identity to assessment of situations and threats. To clarify these roles and layers, the model in figure 4.1 was presented.

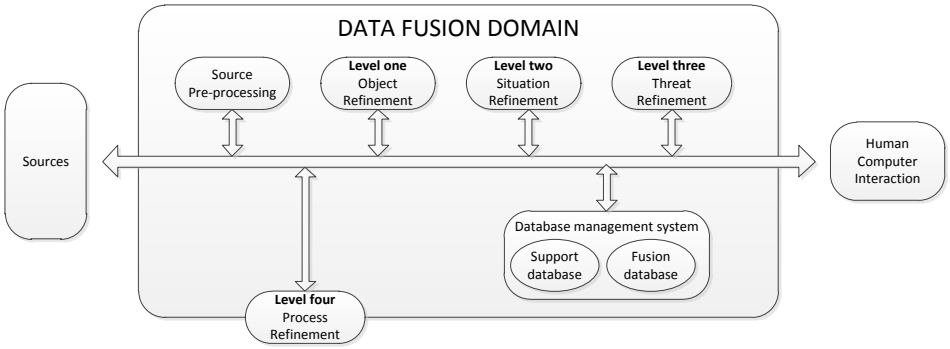

Figure 4.1: The JDL data fusion in its original form, published by the JDL working group for Data Fusion in 1987.

The model prescribes four levels of data fusion, which mature the perception of measurements on signal and cognitive levels to present to human decision makers. Although the JDL Fusion Model has been the most accepted and used data fusion model, it also faced some criticism. Especially that the model is a canonical guide to a fusion architecture and the description suggests that each level builds on the previous level, so that an architecture of complete assessment are necessary to deal with higher level fusion, and that each layer worked independently from other layers. The model also had a focus towards military applications and used a lot of terminology, such as ”threat refinement”, who brought a focus on for example tactical targeting applications and made transfer of the concepts to other domains not so obvious.

Steinberg et al. (1999) presented a revision to the JDL model, which matured the initial model to be more appropriate for a research community, rather than strictly military driven research. The revision II model introduced a better methodology for interplay between layers, a level 0 fusion layer for sub-object fusion and the notion of estimating perceptual states instead of just physical objects. Figure 4.2 illustrate the revised JDL fusion model by Steinberg et al. (1999)

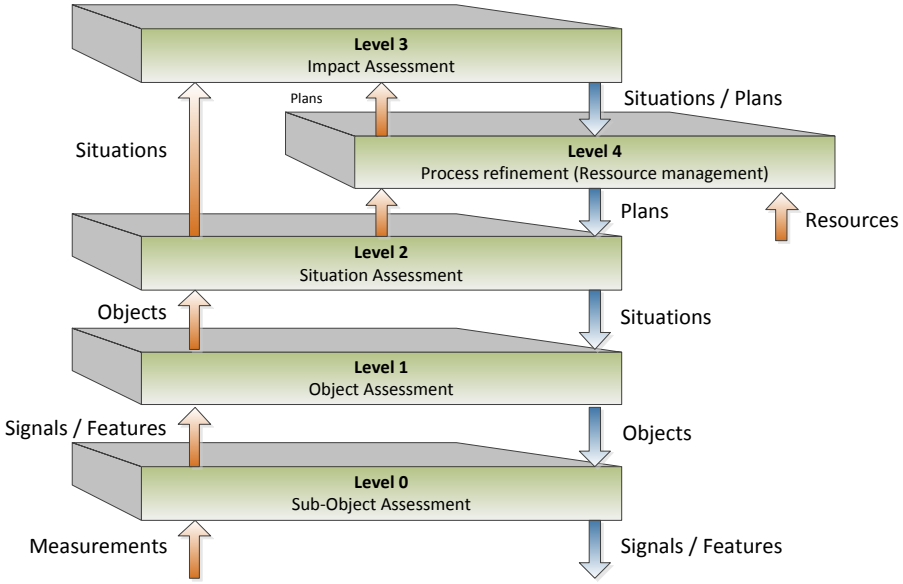

Figure 4.2: Revision 2 of the JDL model by Steinberg et al. (1999), presents a better interplay between fusion layers and a more loose coupling of the architecture.

So far, I have omitted a more formal description of the layers in the JDL Data Fusion Model. This was mainly to avoid the original and military centric definition and to focus the effort towards the modern and more suitable revised model by Steinberg et al. (1999). The layers defined in the revised model are described as following:

Level 0 fusion delivers assignments which include signal detection on the basis of integration of a time-series of data (e.g. the output of an A/D converter) and feature extraction (e.g. from images in a machine vision system).

Level 1 fusion assignments involve associating metrics (or tracks from prior fusion nodes in a processing sequence) into association hypotheses.

Level 2 fusion involves a process of associating tracks (i.e. hypothesized entities) into aggregations. The state of the aggregate is represented as a network of relations among its elements. The revised model admits any variety of relations to be considered - physical, organizational, informational, perceptual - as appropriate to the given system's mission. As the class of relationships estimated and the numbers of interrelated entities broaden, the community tends to use the term *situation* for such an aggregate object of estimation.

Level 3 fusion is usually implemented as a prediction, which can draw particular kinds of inference from Level 2 associations. Level 3 fusion estimates the *impact* of an assessed situation; i.e. the outcome of various plans as they interact with one another and with the environment. The impact estimate can include likelihood and cost/utility measures associated with potential outcomes of planned actions.

Level 4 processing involves planning and control, not estimation. Some discussion points towards there being a formal duality between estimation and control, as there is a similar duality between association and planning. Therefore, level 4 assignment involves assigning tasks to resources.

Although this revised definition of fusion levels also might seem a bit canonical, where each layer relies on estimates from the preceding layer, it is noted by Steinberg et al. (1999) that data types are not restricted to any layer, and that one methodological approach can contain elements from several or even all layers.

Acknowledging the revised JDL model, Llinas et al. (2004) presented a thoughtful revisit to the model. Along with a great deal of fine tuning of the semantics in the revised JDL model, Llinas et al. (2004) also proposed to remove level 4 processing from the data fusion model. A key argument is that level 4 is defined as a resource assignment and planning layer. I agree that planning and resource management is an essential part of any autonomous system, but I do not see it as a part of the information fusion system. System goals and plans are an important part of making a proper impact assessment, but the planning process is not an integral part of the perception hierarchy.

4.1.2 Situation Assessment

Level 2 Information Fusion was initially described by the JDL Data fusion working group as Situation Refinement and was originally considered a process of converting information such as a ballistic missile track to a refined situational picture such as ”a missile is inbound” and thus creating situation awareness among human decision makers, whom should take countermeasures. Within human factors research, the process of achieving situation awareness is defined as situation assessment and thus it was natural to rename level 2 fusion to situation assessment in the revised JDL Data Fusion model.

Several software modules and systems for situation assessment have been presented to the Information Fusion community with various approaches in regards of methodology and application. The research group around Prof. Moses Sudit at the University of Buffalo New York and associated research centers has done numerous works in situation assessment, especially within the domain of cyber security. Sudit et al. (2007) presents the software framework INFERD (INformation Fusion Engine for Real-time Decision-making), an adaptable information fusion engine which performs fusion at levels zero, one, and two to provide real-time situational assessment.

The INFERD framework is considered to be the most elaborate framework for multi-level fusion and situation assessment publicly presented. INFERD uses a graph-based process to hierarchically organize level 0 (L0), level 1 (L1) and level 2 (L2) fusion processes. In the L0 fusion stage system inputs are transformed from heterogeneous data (values, text strings or file based information) to an abstracted data type and represented in a feature node, which are organized in a feature node tree, as understood in basic graph theory. Nodes in the feature tree will become asserted to a binary 1, when a discrete input value matches an

assigned constraint such as equals, less than, larger than, string equality or regular expression match. Once a feature node changes assertion state, the L1 fusion takes over. The top level node of the feature trees is named a Template Node. Rather simplified, an Ordered Weighted Average (OWA) method is applied to calculate the probabilistic 'credibility value' of the template node, based on the number of asserted feature nodes among the children (feature nodes) of the template node. For L2 fusion Sudit et al. (2007) proposes two different methods, either based on concepts from statistical mechanics or entropy.

The INFERD approach to L0 and L1 fusion uses an interesting graph-based method. It forms relationship graphs and uses it to fuse multiple evidence into probabilistic assessments of the reliability of a fact, represented in a template node. A weakness of this approach is, that the method evaluates reliability based on how many indicators point towards that fact being true. In domains like cyber security and airport security control, assessments can be formed on basis of sums of discrete evidence. In robotics, there is often only one sensor associated to a given measurement, but it has inherent uncertainty associated to it. Fusion of multiple sensors very often requires a model driven approach to form L1 estimates and not just a sum of a number of L0 assertions. The L2 fusion only relies on spatial relations between template nodes in the template graph, but as derived in chapter 3, the temporal aspect is essential to assess situations for mobile robots.

4.2 Situation Awareness

A terminology closely related to situation assessment is Situation Awareness (SA). The first adoption of the term was by the USAF (US Air Force) when analyzing the fighter crews returning from Korea and Vietnam, who identified having good SA as a key factor in air combat, or what they called the "ace factor". Performance in air combat was a matter of observing the moves of the opponent and being able to anticipate the next move, before the opponent anticipated your next move.

In the 1990s the term was adopted by the community of human factors research. Human factors science are the science of understanding of the properties of human capability, and defines the concept of human factors as the sets of human-specific physical, cognitive, or social properties which may interact in a critical or dangerous manner with technological systems, the human natural environment, or human organizations. In human factors engineering, these properties can be taken under consideration in the design of ergonomic human-user oriented systems.

Many approaches and models for SA have been proposed within the human factors community, but the definition by Endsley (1995) has been widely accepted and adopted:

> *The perception of elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future. (Endsley, 1995)*

This definition has not only formed a reference within human factors science, but also in other communities who strive towards achieving this type of knowledge such as Information Fusion. A cornerstone in human factors science and engineering is to integrate optimal interaction between human decision makers and computer-based systems, such as military information systems, air traffic control systems, or automation systems. Typically these systems have an enormous amount of information available and a similar amount coming in from information sources constantly. In order to make the human decision makers capable of quickly comprehending complex dynamic situations from large amounts of data, SA systems extract important or critical elements and relations in order to present them in the system interfaces and accelerate decision making.

The relation between situation awareness and situation assessment has been subject to some confusion, as they are applied in many formal and informal fields of research and application. Endsley (1995) establishes the relationship by arguing that:

> *it is important to distinguish the term situation awareness, as a state of knowledge, from the processes used to achieve that state. These processes, which may vary widely among individuals and contexts, will be referred to as situation assessment or the process of achieving, acquiring, or maintaining SA.*

In short terms it defines situation awareness as a state of knowledge whereas situation assessment is the process used to achieve that knowledge.

4.2.1 The Endsley model of Situation Awareness

The need for SA is critical in a number of environments, such as Air traffic control, large automation systems control, and tactical and strategic systems (also for civilian use). But also many every day activities need a dynamic update of SA to ensure correct decision making and effectiveness, such as walking, driving in heavy traffic or operating heavy machinery.

Acquiring and maintaining SA becomes increasingly difficult when complexities and dynamics of environments increase. In these environments, many decisions are required in a fairly narrow window of time and many tasks require a continuous and up-to-date environment analysis. For human-in-the-loop systems, the challenge of maintaining SA can range from trivial to require a major part of the operator's time and attention.

Researchers who analyzed human decision processes in a wide range of application areas do agree, that expert decision makers begin their decision process by classifying and understanding a situation, before immediately selecting an action strategy (Endsley, 1995). What is interesting in that conclusion is that if situations are classified to known situation models, actions can often be chosen immediately without much deliberation involved in deriving the solution.

Evidence points towards, that a basic human process is forming the achieved SA picture of the current situation and matching it towards prototypical situations in memory. Furthermore, these prototypical situations are considered to be mentally mapped to corresponding appropriate actions, making it efficient to adopt a course of action when the situation is known (Klein, 1989). The strategy is also known from medical decision making, where some emergency procedures are being formulated as Standard Operating Procedures (SOPs), where given situational patterns can be mapped to a list of appropriate actions in times where time is too critical for elaborate deduction of courses of action. In his studies of ground fire commanders, Klein (1989) found that conscious deliberation of action strategies was rare. In most of the cases, experts focused their attention to classify the situation and hereafter immediately yield the solution strategy from memory.

On basis of these observations and the basis of situation awareness, Endsley (1995) formulated a model for SA and how it relates between the human mental model, goals and the system/task environment where SA is derived from. The model is illustrated in figure 4.3.

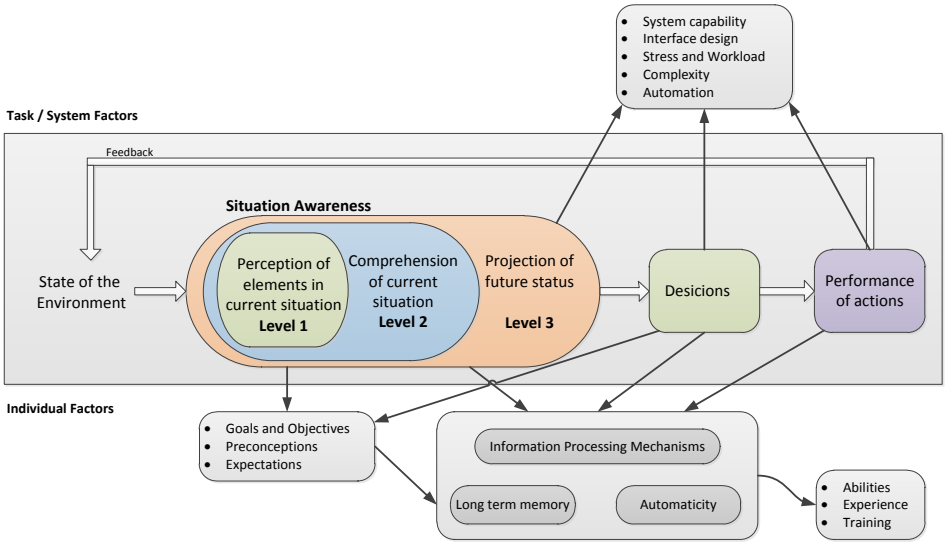

Figure 4.3: Endsley's model for situation awareness in dynamic decision making. The core feedback model is illustrated in the center and is augmented with the relations to human cognition factors for decision making.

In the model, Endsley defines three levels of SA. The levels are organized in

a hierarchical structure, who defines how good a level of SA has been acquired by the human. The three levels are defined by:

Level 1, Perception of Elements in the Environment.
This first step for acquiring SA is to actually identify and perceive the relevant elements of the environment, including attributes such as position, dynamics and status. An automation system operator needs to know the status of his machines, the parts and the backlog in order to form any awareness of the current situation.

Level 2, Comprehension of the Current Situation.
When comprehending a situation, a human fuses the disjoined Level 1 elements and synthesizes the situation. Level 2 SA goes beyond just being aware of elements in the situation but includes understanding of the significance of these elements seen in the light of the operator's current goals. Examples can be automation system operators, who often must piece separate pieces of information, such as measurements and status readings, as well as deviation from expected readings to form an understanding of the current situation.

Level 3, Projection of Future Status.
The third and highest level of SA is formed by being able to project future actions and status of the environment. Through knowledge of status and dynamics of elements in the situation (both Level 1 and Level 2) an operator can be able to predict the near future behavior of a system and perform actions to achieve his goals in the most favorable manner, and hereby avoid critical situations instead of entering the crisis and being forced to react upon it and resolve it.

Some of the objectives within human factors research and higher level information fusion do overlap. Human factors research and engineering often focus on creating systems, which have the optimal combination of human cognition and system processing capabilities by optimizing the information for human operators. Higher level information fusion research and engineering is often focused on designing systems that fuse multiple information sources and can generate higher level information and estimates to human operators.

Addressing the information fusion community, Endsley (2011) presented a discussion of the couplings between Cognitive engineering and the Information Fusion Problem. A model which maps the two community models together is presented in figure 4.4.

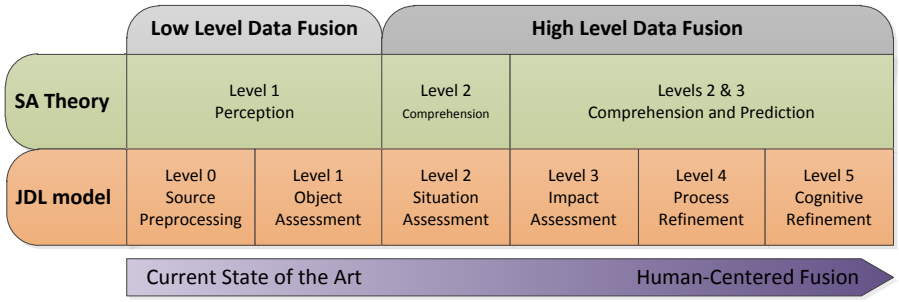

Figure 4.4: Mappings between the SA Theory model by Endsley (1995) and the revised JDL model for Data Fusion by Steinberg et al. (1999).

It is clearly seen in figure 4.4, that there is a match between theories from the two communities. The most significant difference is the focus and approach to the problem. Information Fusion is born from the fusion technologies in the domain of fusing several sensors and real-time sources into better or more informative estimates. Human factors cognitive engineering is born from making ideal systems for human operators and thus has been forced to focus on information which must be presented to the operator.

4.3 Modeling Spatial and Temporal Relations

Already in the initial discussion about situations in chapter 3.2 on page 40, it was obvious that from a human descriptive point of view, a situation must be considered as an aggregation of elements from the environment, not only in metric and spatial aspects but also in the temporal domain. Within his work on situation calculus, Reiter (1991) models a dynamic world as progressing through series of situations as result of actions being performed within this world. In this definition a situation is not a state, but rather the explicit sequence of states as underlined by Reiter (1991) as follows:

> *A situation is a finite sequence of actions. Period. It's not a state, it's not a snapshot, it's a history. (Reiter, 1991)*

In domain of Information Fusion, it is not explicitly defined that there is any evaluation of temporal relations involved in situation assessment. The JDL model defines situation assessment in a domain as forming a "*network of relations among its elements*", which does not prescribe any temporal relations. In the INFERD situation assessment framework, described in section 4.1.2 on page 57, situation prototypes and models are matched without modeling the temporal relations of the data structure.

A single observation can be used for a multitude of purposes in a robotic system. Distances can be measured, objects can be recognized, and motors can

be controlled. However, neither of these observations describes a situation, but maybe a significant state in a situation. Figure 4.5 illustrate a plot of data from a laserscanner mounted on a mobile robot platform, which captures the contours of 5 pairs of human legs.

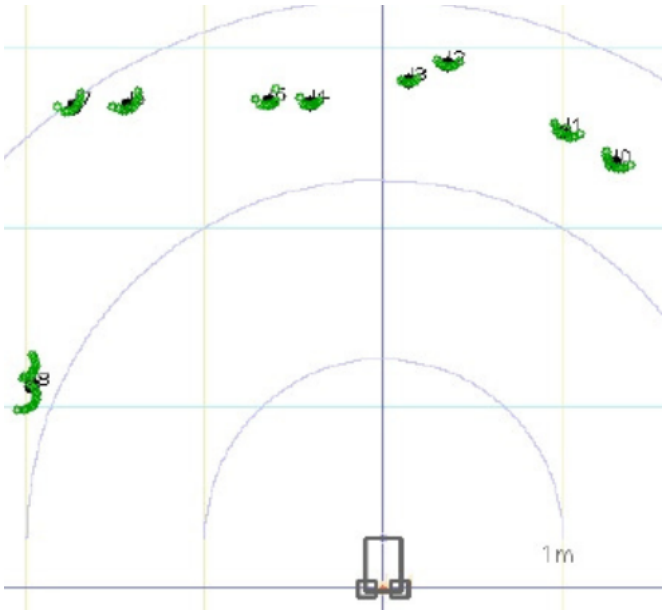

Figure 4.5: A number of human legs detected and recorded using a laserscanner at floor level on a mobile robot platform. Using only this single instance of information, it is impossible to have any comprehension of the situation and to do any prediction of future events.

Observing the leg contours in figure 4.5, it is impossible to grasp what the persons on the scan are doing, in what direction they are going, or at what velocity they are moving. Using the temporal domain to connect sequences of measurements, enables us to comprehend what is actually going on in this sequence and maybe even predict future status.

In their work on using an ontology driven approach to formalize reasoning around situation assessment in disaster environments, Little and Rogova (2005) proposes to integrate the temporal aspect by the use of two separate ontologies. By their approach, situations are characterized by spatial items of interest in the SNAP ontology in different levels of granularity. Temporal items of interest are defined in the SPAN ontology which also is used to define the relations between SNAP entities. Although I very much agree, that it is a good idea to separate description models to simplify modeling and motivate re-use, Little and Rogova (2005) becomes challenged when defining the SPAN temporal ontology. Their disaster ontology example for SPAN becomes mixed between process elements and purely temporal elements, where I will argue that elements such as `Ambulance -> Pick-up` and other process entities, is not an explicit temporal element.

4.3.1 Methods and assumptions

In order to design a framework to organize data from a mobile robot in both spatial and temporal domains, a suitable modeling framework must be selected. Many computational approaches have been proposed, such as rule-based methods, graph-based methods, case-based reasoning, hidden Markov models, multi-agent systems, neural networks, situation calculus and many hybrid methods especially involving probabilistic additions.

In chapter 3 it was outlined, that important situational patterns can be constructed from a number of sources, either learned from good or bad situations. These patterns could maybe be partially hand-crafted by experts to capture interesting key-points. However, in dynamic environments, it is evident that not all situational patterns can be manually crafted. Critical characteristics can often be found in un-modeled uncertainties or other elements which is not a core part of the perception and control of a system, which concludes

- The model must be definable by experts but also be able to build or refine itself dynamically to comprehend real-world dynamics.

Not all models are capable of representing both spatial and temporal relations. Graphical models are very popular for this purpose, as they are very well suited to structure and represent relations among elements and generally cover two important components:

- The structural component, the graph (directed, undirected)
- The quantitative component (probability, possibility, belief)

Most variations of the graphical methods use a variation of a probabilistic framework to combine the atomic structure in the graph-based model with the aspects of uncertainty. Another general benefit for the graphical methods is that they are easily visualized and there is a large body of efficient algorithms and software packages available. As for all methodologies, graphical methods also have their drawbacks. In general, the methods are particularly good for sparse networks as in general methods of inference are NP-hard. In an open and dynamic world, it is often difficult to fix the variables and structure, as well as handling dynamic situations adds complexity of the model (Rogova, 2011).

4.3.2 The Extensible Markov Model

The framework for modeling the spatio-temporal characteristics of situations has been selected to be the Extensible Markov Model (EMM), originally presented by Dunham et al. (2004) as a tool for on-line analysis of the spatio-temporal structure of data sets. Markov models are a popular data modeling tool for spatio-temporal problems, as the graph-based structure fits intuitively too many natural processes and is easily visualized. The traditional structure of a Markov Chain is a static representation of a number of states in a graph and a number of

edges connecting elements of the graph with associated transition probabilities describing the probability of traversing the edge from one state to another.

The static properties of the Markov Chain only make it suitable for examination and evaluation of static data sets, where on-line and dynamic datasets does not fit. With most real-world on-line data sources, such as robotic sensor systems or perception algorithms, data values change over time, resulting in these typical problems using a static Markov Chain for real world data:

1. The required structure of the Markov Chain may not be certain at the model construction time.
2. As the real world being modeled by the Markov Chain changes, so should the structure of the Markov Chain.

To deal with this problem, Dunham et al. (2004) proposed the Extensible Markov Model (EMM). The EMM is in essence a time varying Markov Chain, with the capability to learn new states dynamically and adjust its structure. Furthermore, the edges are used for modeling state transitions and the associated probabilities based on the input data to the model. Dunham et al. (2004) experimentally verifies the model using examples from prediction of river flow and prediction of traffic volumes for both computer networks and roadways.

In work on a similar problem Goldberg and Mataric (1999) have presented the Augmented Markov Model (AMM). The AMM also learns new states when new data are seen in the model and adjusts transition probabilities according to the number of transitions along each edge compared to the total number of transitions from that state. Unlike the EMM, when the AMM is constructed after the learning phase, it is frozen and no longer being updated in the prediction phase.

The EMM model uses a similar approach to AMM, but offers a great deal more flexibility with the following key differences (Dunham et al., 2004):

- EMM can continue to learn during the application (prediction, etc.) phase.
- The EMM is a generic incremental model whose nodes can have any kind of representation.
- State matching is determined using a clustering technique (as further discussed in chapter 5).
- EMM not only allows the creation of new nodes, but deletion (or merging) of existing nodes. This allows the EMM model to ”forget” old information which may not be relevant in the future. It also allows the EMM to adapt to any main memory constraints for large scale datasets.
- EMM performs only one scan of new data and therefore is suitable for online processing of streaming data.

The construction process on an EMM is continuous throughout the life of the model. New data is evaluated against the existing nodes using clustering to deal with measurement inaccuracy and noise and is either matched to the existing model or added as a new node. Figure 4.6 illustrates the update process of an EMM with a new set of data.

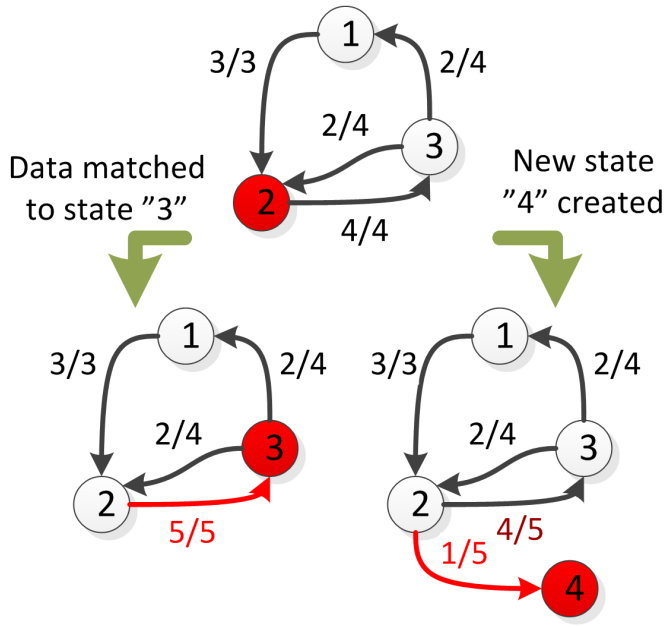

Figure 4.6: Simultaneous construction and matching in the EMM model. A new data set can either be matched an existing node or create a new node if there is no similarity to any existing nodes.

4.3.3 Other frameworks for spatio-temporal modeling.

Another popular approach for modeling spatio-temporal relationships in a graph-based setting is using Hidden Markov Models (HMM) (Rabiner, 1989). A HMM is a statistical extension to the traditional Markov Model with a set of unobserved (hidden) states. The separation between the measured states and the output states in the HMM gives an advantage when dealing with uncertain and noisy data. HMMs are a method of choice in several applications of temporal pattern recognition, such as speech, gesture, or handwriting recognition.

In their work in recognizing situations in vehicular scenarios, Meyer-Delius et al. (2007, 2009a,b) develops a framework to apply HMMs to recognize situational prototypes. In their work, they hand-craft a number of situation recognition HMMs, each representing a known situation to be recognized. In the presented vehicular scenario, they defined three distinct situations to recognize: *Car passing*, *car following* and *car aborted passing*. Each HMM was trained with a large volume of manually annotated data, which enabled it to robustly recognize the situation patterns.

Although the HMM is a significantly more advanced and complete probabilistic framework, that is also its weakness. There is a significant amount of

thoughtful modeling in designing the suitable models, as both the "observation" state space and the hidden states and their relations must be hand crafted. Furthermore, all probabilistic matching between input data and the HMM is done internally in the trained model which makes it very difficult to debug, tune or optimize parameters, especially for non-experts.

4.4 Situation Assessment Modeling Framework

As I have described so far in this thesis, there is a fair body of work related to situations, their definition, their description, their recognition and how humans perceive them. Approaches for technical situation models span from complex probabilistic models to specific predicate logic dedicated to situations. Although none of them specifically addresses robotics from a situation assessment context.

In this section I will describe the choice of situation modeling framework used to represent the situations for mobile robots. Robotics in general is a very cross disciplinary field, building intensively on state-of-the-art from areas like control theory, signal processing, artificial intelligence and mechanical engineering. The broad landscape of research disciplines, all contributing to solve the challenge of making mechanical computerized systems solve real-world challenges, makes the information within a robotic system highly heterogeneous. Common data types which need to be considered are:

- Raw sensor readings, e.g. from distance sensors, encoders or analogue inputs.
- Simple positions, e.g. from dead-reckoning or from position servos.
- Pose estimates of robot and objects, e.g. from localization algorithms and object trackers.
- Status values, e.g. from hardware or software components.
- Logic values, e.g. from rules monitoring processes.
- Process details, e.g. details from the planning system about current mission and task.
- Symbolic information, e.g. from an AI or inference layer.

Using the first order Markov chain nature of EMM, the spatio-temporal nature of situations can be represented together with transition probabilities, i.e. without the complexity of hidden states in a HMM. States in the Markov chain are a unique set of information or circumstances at one given time. We define a situation instance as the sequence of one or more of these states, who can be translated to meaningful knowledge about the state of a dynamic environment.

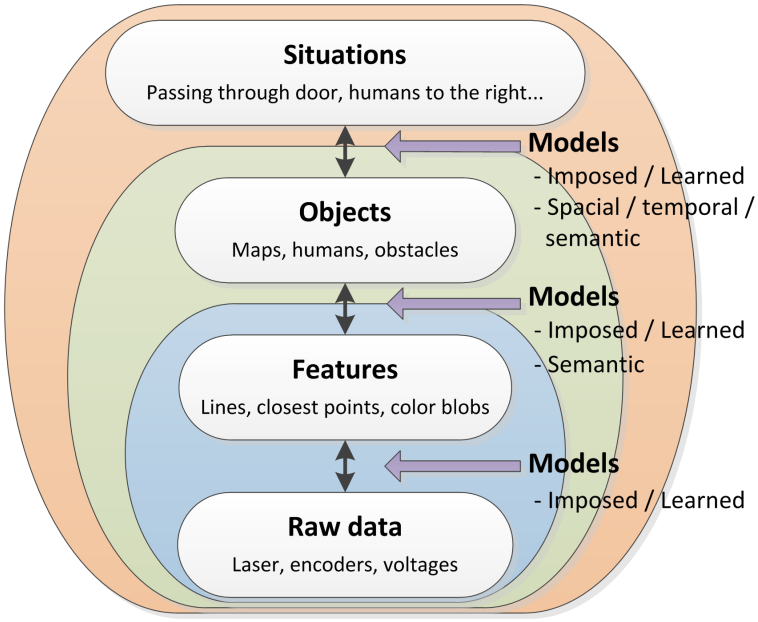

Figure 4.7: Perception information for mobile robots is often organized in a hierarchy in the robot control architectures. Situations combine these layers in the spatio-temporal dimension.

Situations add an additional level of abstraction to traditional spatio-temporal models, as both symbolic and sub symbolic information can be fused to form the situation states and combine them in the temporal and spatial dimension. Figure 4.7 illustrates the common understanding of a mobile robot perception hierarchy where situation models are placed in the highest level, as an assessment can involve data from all preceding layers, as also emphasized by Steinberg et al. (1999) in the JDL model.

To embrace this, a situation model must be parameterized by any combination of data sources available in the heterogeneous perception hierarchy, which exists in the frameworks of modern mobile robotics.

Following Endsley's model for Situation Awareness (Endsley, 1995), the full benefit of achieving SA is the ability to both comprehend the current status of the environment as well as being able to predict future status and events, as outlined in Endsley's SA hierarchy in figure 4.8. A comprehensive situation model should be capable of covering the three levels of situation awareness.

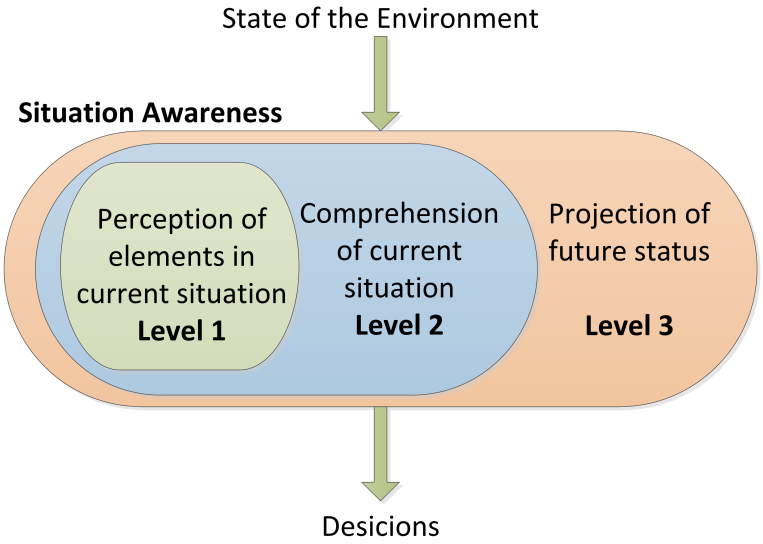

Figure 4.8: Endsley's models of Situation Awareness define three levels of awareness.

Using the EMM for situation representation, it can be considered to have three similar layers of abstraction, which are a natural part of the Markov chain approach.

Level 1 SA concept of the model
Is the definition of what data that should be included in the model. Data selection must be done carefully, as too many parameters can cloud the observed structures and too little might give inconclusive observations.

Level 2 SA concept of the model
Is represented in specific state-sequences which form a situation. These models must be capable of being pre-imposed from expert descriptions or classified from learned data during runtime.

Level 3 SA concept of the model
Is the evaluation of transition likelihoods from the current state to other defined states. These models are used for prediction of future error states and inference i.e. for triggering counter measures.

Comparing the layers to the revised JDL model for Data Fusion, there is a reasonable resemblance between this model definition and the comparison illustrated earlier in figure 4.4 on page 62.

In the work presented in Beck et al. (2011), who introduced this modeling approach, we focus primarily on designing a level 1 model and show through experiments that it can be used to learn meaningful level 2 situation models and level 3 prediction models. In the experimental work in chapter 7 on page 109 I investigate further how all three levels of situation awareness can be achieved as a result of situation assessment using this model.

4.4.1 Model definition

To represent a state in a situation, a set of objects A was proposed by Barwise and Perry (1981) as the parameters which form a situation entity. The parameterization model, here defined as s, defines which data-components are included in the situation assessment process. s is forward in this thesis referred to as the *situation model* and the parameterization process is referred to as *situation model design*. The situation model s is defined in equation 4.1 as

$$s = \{d_1, ...d_n\} \tag{4.1}$$

The situation model s is parameterized by a number of n data-parameters d_n which are used to form the situation assessment.

In the situation assessment process, a number of states are recognized as instantiations of the set of objects s at certain space or time. The known set of states is represented in a state space S with a set of m states at a given time, as defined in equation 4.2.

$$S = \{s_1, ...s_m\} \tag{4.2}$$

Temporal relations are represented through the Markov Chain structure as a number of edges which describe the possible transitions between states s_n in the state space. At each transition between states, the count of transitions between states is updated and represented in the state transition matrix N in equation 4.3.

$$N = \begin{vmatrix} n_{00} & \cdots & n_{i0} \\ \vdots & \ddots & \vdots \\ n_{0j} & \cdots & n_{ij} \end{vmatrix} \tag{4.3}$$

Where n_{ij} represent the number of transitions from state s_i to s_j .

Transition likelihood from state s_i to s_j are maintained on-line by computing the number of outbound transitions from s_i to s_j in relation to the total number of outbound transitions from s_i, given by

$$L(s_t = s_j | s_{t-1} = s_i) = \frac{n_{ij}}{\sum_{z=1}^{m} n_{iz}} \tag{4.4}$$

and represented in the transition likelihood matrix A in equation 4.5, where a_{ij} represents the transition likelihood from state s_i to s_j.

$$A = \begin{vmatrix} a_{00} & \cdots & a_{i0} \\ \vdots & \ddots & \vdots \\ a_{0j} & \cdots & a_{ij} \end{vmatrix} \tag{4.5}$$

Valid transition likelihoods are important for predicting occurrence of future, important states and hereby approach Level 3 Situation Awareness. The on-line calculation of transition likelihoods is utilized for situation transition prediction by the proposed prediction engine in section 4.5.

State matching is based on a data clustering, where a model independent clustering algorithm determines the state relationship of measurements, instead of learning data association into the models as with Hidden Markov Models. The clustering is further elaborated in chapter 5 on page 77.

4.5 Situation prediction engine

This section presents a proposed engine to perform on-line prediction calculations from the current node of the EMM model to any other node in the model. By calculation of the maximum likelihood sequences through the EMM graph, the most likely transition sequence between two nodes can be calculated together with the likelihood of this sequence occurring. Using this knowledge, robot control systems can prepare for the most likely future events and possibly activate countermeasures if likelihoods of entering critical situations become unacceptably high.

Thus, the goal of this section is to design a methodology to keep an updated maximum likelihood estimate of transitioning from the current node to any other nodes in the EMM graph, to enable monitoring of possible critical situations and improve the avoidance of these situations instead of recovery from them.

In the EMM model, transition likelihoods are annotated in the description of each edge connecting the states s_i and s_j by equation 4.4 on the facing page. Considering the EMM as a traditional graph, the transition likelihoods can also be considered as (inverse) weight factors of traversing the edges.

Well known algorithms exist for processing and searching graphs with weight factors assigned to the edges, where the most famous is Dijkstra's algorithm. The algorithm was presented by the Dutch computer scientist Edsger Dijkstra in 1959 and is a graph search algorithm to solve the shortest-path problem in a graph from a single source node.

The Dijkstra algorithm iterates through and "visits" all nodes connected to the start node and assigns the lowest cumulative cost to each node. This continues to iterate through nodes connected to the "visited" nodes, until the end node is reached or until all nodes have been visited. As a result of the algorithm is a route-map of lowest possible cost sequence from the start node to any visited node in the graph, as well as the cumulative cost of following this sequence.

In general, the characteristics of Dijkstra's algorithm fit fairly well with the desired prediction calculations, but instead of calculating the lowest cost, I am interested in extracting the maximum likelihood of the sequences.

4.5.1 Maximum likelihood of graph sequences

Following the chain rule of probability, it is possible to calculate the joint distribution in terms of transition probabilities of the set X_{seq} using the conditional probabilities. Consider the sequence of nodes

$$X_{seq} = \{s_1, ..s_n\} \tag{4.6}$$

To calculate the conditional probability of this sequence, the definition of conditional probability can be applied (Russell and Norvig, 2010, p. 512)

$$P(s_n, \ldots, s_1) = P(s_n | s_{n-1}, \ldots, s_1) \cdot P(s_{n-1}, \ldots, s_1) \tag{4.7}$$

And repeating the deduction on the final product of equation 4.7 the conditional probability of the joint distribution of the set X_{set} is given by the product

$$P(s_n, \ldots, s_1) = \prod_{i=1}^{n} P(s_i \mid s_{i-1}, \ldots, s_1) \tag{4.8}$$

4.5.2 Modification of Dijkstra's algorithm

Replacing the cost minimization step in Dijkstra's algorithm with a maximization of this joint distribution of conditional probability calculation makes Dijkstra's algorithm a maximum likelihood estimator, instead of shortest path planner.

It is implemented into the algorithm by modifying the original core loop which is illustrated for comparison in pseudo code in listing 4.1.

```
1 //Q is the set of non-visited nodes
2 while Q is not empty:
3  u = vertex in Q with smallest distance in dist[] ;
4  remove u from Q ;
5  if dist[u] = infinity:
6   break ;
7
8  for each neighbor v of u:
9   remove v from Q.
10  alt = dist[u] + dist_between(u, v) ;
11  if alt < dist[v]:
12   dist[v] = alt;
13   previous[v] = u;
14 return dist;
```

Listing 4.1: Pseudo code of the core loop in Dijkstra's algorithm for solving the shortest path problem.

Listing 4.2 illustrates pseudo code of the modified core loop, where the main changes are

- Line 3: `u` is now the vertex with largest likelihood.

- Line 10: Instead of calculating the sum of distances, the product of likelihoods is calculated.
- Line 11: Max likelihood is updated for each node instead of min distance.

```
1 //Q is the set of non-visited nodes
2 while Q is not empty:
3  u := vertex in Q with largest likelihood in
       maxLikelihood[] ;
4  remove u from Q ;
5  if maxLikelihood[u] = 0:
6   break ;
7
8  for each neighbor v of u:
9   remove v from Q.
10  alt = maxLikelihood[u] * likelihood_between(u, v) ;
11  if alt > maxLikelihood[v]:
12   maxLikelihood[v] = alt ;
13   previous[v] = u ;
14 return maxLikelihood;
```

Listing 4.2: Pseudo code of the modified Dijkstra algorithm for solving the maximum likelihood estimation problem.

If the only purpose was to calculate the maximum likelihood between a start node s_1 and a specific end node s_n, the algorithm could be terminated at line 4 if `u` $= s_n$. The resulting maximum likelihoods to any node s_n in the EMM graph can now be extracted from the `maxLikelihood(`s_n`)`. The sequence which leads to this likelihood can be extracted by iterating backwards through the `previous[v]` from s_n until s_1 is returned (and then reversing the sequence).

Figure 4.9 illustrates an example of the maximum likelihood calculations between node `N1` and `N6`. Transition likelihoods are noted on each edge and the calculated maximum likelihood in each node is noted besides the node id in `()`.

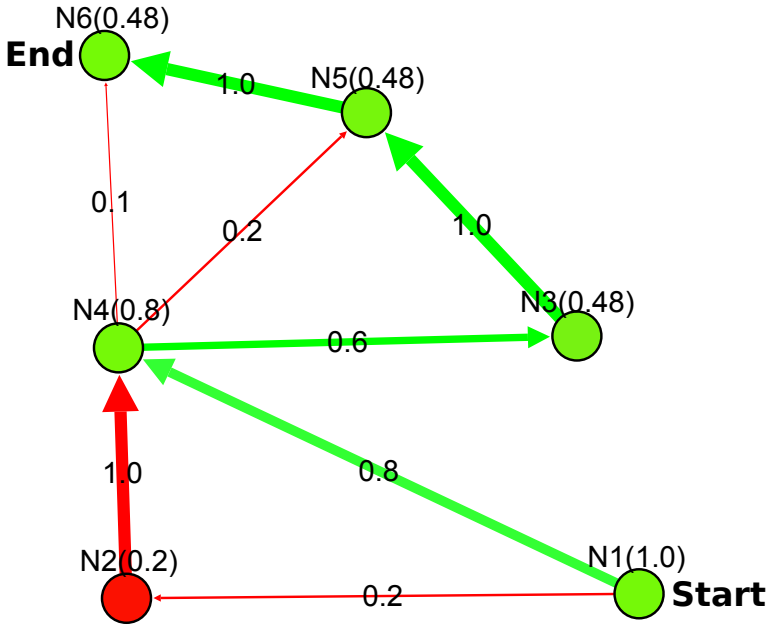

Figure 4.9: Example of maximum likelihood calculations using the modified Dijkstra algorithm. The green edges illustrate the route of maximum likelihood with transition probabilities noted on each edge. The maximum likelihood for each node is noted in () besides the node id.

The algorithm is implemented in a prediction engine module which continuously updates the maximum likelihood map, when the EMM model is changed. Experiments in chapter 7, and in particular section 7.4 on page 132, investigate the usefulness of using this module for prediction of future status of the situations described in the EMM model.

4.6 Summary

This chapter provides an overview of the areas of research which have dealt with situation assessment and situation awareness. As it is clearly evident, these concepts have been addressed from many sides of research, where the most dominant have been within information fusion and human factors research.

Already in 1987 situation assessment was defined as an essential part of the information fusion domain, as the US Army Joint Directors of Laboratories (JDL) working group within data fusion presented the model, known as the JDL Data Fusion model, which clearly identified the level 2 information fusion process as situation assessment. The model and research of that age was clearly directed towards military purposes, and the examples discussed were identification of relations in battlefield overviews and formation of higher level knowledge from multiple instances of target tracking. Modern research within information fusion has addressed situation assessment from other contexts than military, such as cyber security. Although a number of good approaches to situation assessment have been proposed, the techniques have not yet found their way into large scale

deployment in real-world systems. Situation assessment from an information fusion perspective has yet to be addressed within the domain of robotics.

Again from a military background, the research within human factors science has also targeted the concepts of situation awareness and situation assessment. Analysis of how humans form mental models of situations have successfully been applied in cognitive engineering for designing human-computer interfaces, which optimally utilizes strengths of both. Massive volumes of data can be interpreted and processed by computers but complicated reasoning and decision making on behalf of the results are still optimally solved by humans. The 3-layered human situation awareness model, presented by Endsley (1995) will form a baseline for the conceptual situation modeling framework I present in this thesis. Compared to the JDL Data fusion model, Endsley's model describes an actual process of cognitively building a model of situation awareness in multiple layers, which all are addressed in the presented work.

The final contribution in this chapter is the definition of the situation assessment modeling framework, which forms the basis for the approach to situation assessment in this thesis. Using the Extensible Markov Model by Dunham et al. (2004) as modeling basis, a framework for modeling situations as spatio-temporal sequences is presented. By the use of this approach, all three layers of situation awareness as described by Endsley (1995) can be achieved and yet be founded in a mathematical model which is suitable for implementation and both human and computerized interpretation.

In order to achieve prediction of future status, a novel prediction engine was designed to process the situation model and continuously calculate the maximum likelihood from current to other states in the situation model. Using this engine makes it possible to monitor the likelihood of entering critical situations and thus prepare certain control algorithms for handling it, before any error occurs to be recovered.

5

Processing Information Streams

The general nature of communication infrastructures in robotic systems are designed from the paradigm that most data sources are continuous, high volume sensors. Besides the high volume sensor information, the results of a level 0 or level 1 fusion process are often estimates, pose, and state values which are lower in volume but often have the same high data rate as the raw sensor data. The continuous streams of information set certain requirements to the performance of the infrastructure, but also to the design of the algorithms which generate higher level information from these sensor data streams.

Research targeted general data processing and computing have recently begun to see a similar challenge in the application they need to address. Many applications, such as web-technology and database systems, could traditionally consider data management and processing as dealing with data that might change over time, but being relatively constant in volume. Now, with dynamic and interactive websites and the possibility to gather information such as click-streams and user preferences from millions of users, the rate and overall volume might be so great that all data cannot be stored for processing, which leads to requirements for efficiency and scalability. In other cases, information volumes might be manageable, but the entire approach of data streams turns what has been a static view of a problem and adds the temporal dimension to it.

The same challenge has appeared in many application areas, such as phone records analysis, financial transactions, data network security monitoring, and automated surveillance. All these applications involve high volumes of data, which are collected through an on-line stream, and just as much information lies in the temporal domain, such as peaks in telephone activity in non-office hours and anomalies in surveillance by detecting persons following unusual sequences of behavior.

Detection of patterns of interest in datasets exists within a number of scientific areas, such as Machine Learning, Statistics, System identification. This chapter focuses on analyzing approaches towards pattern and knowledge extraction from information streams as well as selection of approach and design of the situation assessment data processing module.

5.1 Model-based approaches

A common approach to extracting information from data streams within the control engineering and signal processing community is using a model-based approach. Using models has enormous benefits when using their results in direct computations, as outputs are explicitly defined and internal structures often only change little or to a certain degree. The drawback of most model-based approaches is that they require a complex analytical process, often by a skilled expert to create and parameterize the models.

A wide range of filtering and estimation algorithms has been developed to perform on-line matching between various dynamic models and sensor information, such as the Kalman filter, the particle filter and countless derivations. The goal of these algorithms is to estimate unobservable states or to increase the quality of information in the system and they are generally considered to belong in the Level 0 and level 1 of the JDL Data Fusion Model.

Dynamic Bayesian network approaches, such as the Hidden Markov Model (HMM) is another popular model-based approach towards pattern recognition. Its application towards matching uncertain sensor stream data towards specific situation models has been presented by Meyer-Delius et al. (2009a) as described earlier. HMM is excellently suitable for adapting their parameters from training data, but as their observation structure must be assigned to states of outputs, it becomes difficult to design a system which learns the model structure from data.

A key element in the motivation to use EMM as modeling framework was the fact that the structure of situation models cannot be fully known before operation, and thus requires an information processing framework, which supports on-line and dynamic creation of the model. This requirement criterion generally disqualifies model-based information processing methods.

5.2 Pattern recognition and cluster analysis

Pattern recognition covers a rather large area of signal processing approaches, algorithms and applications. Duda et al. (2001) describes pattern recognition as

> "*the act of taking in raw data and taking an action based on the category of the pattern*"

This is exactly what is intended to for mobile robotics. The aim of pattern recognition is to classify data based on either a priori knowledge or statistical information extracted from patterns submerged in the data. Being an approach in the category of pattern analysis, cluster analysis is considered the most significant approach to unsupervised machine learning. Generally clustering is the problem of finding structure in a volume of unlabeled data or in loose terms as "*the process of organizing objects into groups whose members are similar in some way*".

To gain the benefit of the model-learning nature of clustering, an on-line cluster analysis driven approach has been chosen as data processing framework. Combining on-line clustering with the use of the first order Extensible Markov Model framework covers the gap between the pure clustering methods which only consider the spatial dimension of data, and the pure model-based approaches where most of the model structures can only be designed by hand.

5.3 Definition of information streams

Streams of information have been addressed in a number of different contexts. One definition has emerged in the research community around analysis of massive data-sets. A data stream is considered as an ordered sequence of data-points which can only be read once or a small number of times.

Guha et al. (2000) defines data streams formally as the sequence of points $x_1...x_i...x_n$ read in the increasing order of the indices i. They furthermore state that data stream algorithms must access the input only via linear scans without random access and only require a few (hopefully one) scans over the data. The size of the data-set exceeds any available amount of main memory, so it is not possible for an algorithm to store much of the data. The main novelty of algorithms for handling streaming data is that they can only store a summary model of the data, which can be fitted in memory instead of the entire dataset. This definition of streaming data might apply, if the purpose was to design algorithms which can process large volumes of data, stored on tape-backup systems or other large but finite sources.

Beringer and Hüllermeier (2006) argues that the data stream model follows the basic assumption that some or all data to be processed are not available for random access, but arrive as one or more continuous streams. This is supported by Babcock et al. (2002), whose definition of data streams differs slightly from the models used in the database community:

- The elements of a stream arrive on-line (the stream is "*active*" in the sense that the incoming items trigger operations on the data, rather than being send on request).
- The order in which elements of a stream arrive is not under the control of the system.
- Data streams are potentially of unbounded size.
- Data stream elements that have been processed are either discarded or archived. They cannot be retrieved easily unless being stored in memory, which is typically small relative to the size of the stream.
- Due to limited resources (memory) and strict time constraints, the processing of stream data will usually produce approximate results.

5.3.1 On-line clustering of data streams

To effectively deal with the issues of minimization of I/O and memory cost when extracting useful patterns in very large data-sets, Zhang et al. (1996) proposed the well-recognized Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) algorithm. BIRCH has been designed from a database context, and thus the design focus has been towards creating best possible clustering with a single pass through the data-set, but with the possibility for better results using several passes. BIRCH was the first incremental clustering engine capable of handling outliers (sparse data points which are not a part of the underlying pattern) which are very useful for processing of sensor data. A key strong point about BIRCH is that the algorithm is hierarchical, using a explicitly defined volume of memory to store the cluster tree where the amount of details defined in the leafs of the tree only depends on allocated volume of memory. This hierarchical approach has a possible potential in a situation assessment context, but the BIRCH algorithm is not developed towards the infinite data stream model and would need quite some redesign.

The most common approach towards clustering of data streams is using a variant of the K-means or K-median clustering algorithm. The basic principle of the k-means algorithm is to partition n data points into K clusters, where each data point belongs to the cluster with the nearest mean value. In principle, the problem is NP-hard when applied to a static data-set, but a number of approaches to use heuristics to accelerate convergence.

Beringer and Hüllermeier (2006) use a discrete furrier transform to smooth input data and to ensure real-time performance whilst still maintaining Euclidean distance. The STREAM system, proposed by O'Callaghan et al. (2002), keeps a memory restricted buffer where the data stream is stored. When the buffer is filled, the data-points are clustered into K clusters and only the K centroids are stored. This process is iteratively repeated with new points. Because the K-means algorithm is one of the most simple and intuitive algorithms, its use is particularly common in scenarios where a level of human understanding of the clustering process is required.

The principle K-means algorithm, however, results in data-set being partitioned into K clusters, which assumes the data-set should be partitioned into exactly K clusters, as it is not the case for clustering into a non-finite and dynamic model. In the domain of classification, the k-nearest neighbor algorithm is a commonly applied approach for assigning data-points to one of a number of categories on the basis of an existing and pre-categorized set of data-points. The algorithm is fairly simple, as the new data point gets assigned to a category by majority vote among the k-nearest neighboring data points by means of a given distance metric. The nature of the k-nearest neighbor algorithm is not suited for on-line streaming classification, as all points in the data-set must be analyzed to determine the nearest neighbor, but it does not impose any static limitations on the classification, besides main memory.

5.4 Clustering engine design

The signal input system for the situation assessment framework needs to be capable of handling real-time information streams from robot system frameworks, not in hard real-time but at least keep up with the information flow. The proposed method is a hybrid between K-means clustering and nearest neighbor classification, combining the strengths of both approaches:

K-means clustering has the key benefit of reducing the state-space from the original amount of data-points to a number of clusters, which suits the requirements for information streams.

Nearest neighbor classification is capable of assigning a data-point to the nearest cluster of points, working directly on the reduced clustered state space.

To exemplify the design, consider a system that would monitor the relation between two physical objects, it could be the pose of a mobile robot and its target pose. The pose transformation between the poses is pre-processed within the robotic system and streamed to the situation assessment framework. At the time t, the raw data vector is given by D_t with the dimension n.

$$D_t = \{d_{1t}, d_{2t}, ..., d_{nt}\} \tag{5.1}$$

The existing nodes in the EMM model is defined the list of vectors $S = \{s_1 ... s_m\}$, where m is the number of existing nodes in the model. The initial clustering step is to determine the cluster with the largest similarity J_{max} as described in equation 5.2, using the similarity measure function $J()$, which is elaborated in a following section.

$$J_{max} = \max_{x \in [1,m]} J(D_t, s_x) \tag{5.2}$$

To form the hybrid between clustering and classification, a similarity threshold defined by λ is used to determine if the new data point D_t should be classified as belonging to any existing node s_i in the EMM model or if a new cluster should be created with the center in D_t.

$$D_t \in \begin{cases} s_i & \text{where } i = \underset{x \in [1,m]}{\operatorname{argmax}} J(D_t, s_x) \quad \text{if } J_{max} > \lambda \\ s_{m+1} & \text{thus } m = m + 1 \qquad\qquad\qquad\quad \text{if } J_{max} < \lambda \end{cases} \tag{5.3}$$

The EMM model will initially not hold any states s, unless some sections of the model have been hand-crafted by an expert or a previously learned model has been loaded, so the very first data-set D_1 will naturally be assigned as a node in the EMM model. Through this on-line clustering process, the EMM continuously maintains the temporal transition relations of the model. An illustration of the clustering process is shown in figure 5.1 on the following page.

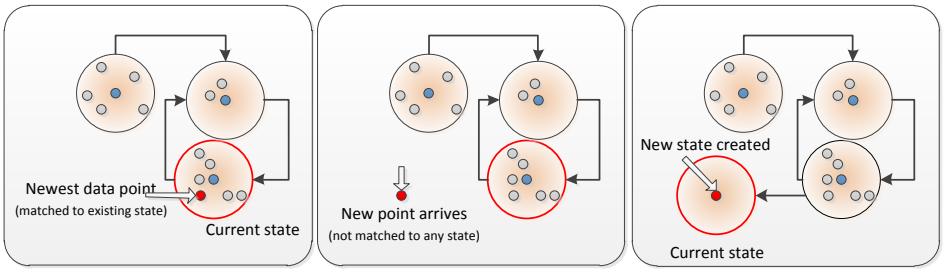

Figure 5.1: The principle of the combined EMM structure and clustering framework. In the example, two new data points are processed in the clustering engine, where the first point is assigned to the current existing node and the second is below the similarity threshold and is assigned as a new node. Note that only cluster-centers are saved, so other points in the clusters are only included for illustrative purposes.

5.4.1 Distance metric

An important component of a clustering algorithm is the distance measure between the data points. This distance metric is used to determine the level of similarity between points in the data space and thus forms the background for associating points to a given cluster. The choice of distance metric will also influence the shape of the clusters, as some elements might be close to each other in one metric and far away according to another metric. A large number of distance metrics exist, where some of the more popular are:

- **Euclidian distance** is the simple geometric distance between two points, given by the Pythagorean formula. The Euclidian distance is invariant to rotation and translations of the data points.
- **Mahalanobis distance** uses a covariance matrix of the data points in the cluster to normalize the distances and thus making it scale invariant.
- **Jaccard similarity** is a measure for comparing the similarity and diversity if sample sets.
- **Cosine similarity** is a measure of similarity between two vectors by calculating the cosine of the angle between them. Cosine similarity is often used to compare documents in text mining applications.

Using the Mahalanobis distance would be a strong choice as metric, as it has the benefit of compensating for scale differences by using the covariance of the cluster-set to normalize. Unfortunately, in order to calculate the covariance matrix of a given cluster, it is necessary to iterate through the actual data points of the cluster, making the metric unsuitable for the proposed clustering approach where only the center point is stored.

The choice of distance metric of the clustering engine has become the Jaccard similarity coefficient, which is defined as the size of the intersection divided by

the size of the union of the sample sets

$$J(s_i, s_j) = \frac{|s_i \cap s_j|}{|s_i \cup s_j|} \tag{5.4}$$

Expressed over scalar vectors, the Jaccard index can be written as

$$f(A, B) = \frac{A \cdot B}{|A|^2 + |B|^2 - A \cdot B} \tag{5.5}$$

The result of the similarity calculation will always be in the range of $J = [0, 1]$ where a value of 1 denotes complete similarity and 0 means no commonality at all. The range of the index makes it significantly easier to assign a threshold λ than using a metric distance, such as the Euclidean, which differs significantly with absolute scale.

5.5 Examining the effects of clustering

As a preliminary evaluation on the usefulness of clustering to extract information from robot sensor data, a simple but relevant example is applied: recognition of motion patterns from raw wheel encoder increments. The sensor input data are chosen as hardware near as possible to illustrate that important higher level information can be extracted, even from very low level data.

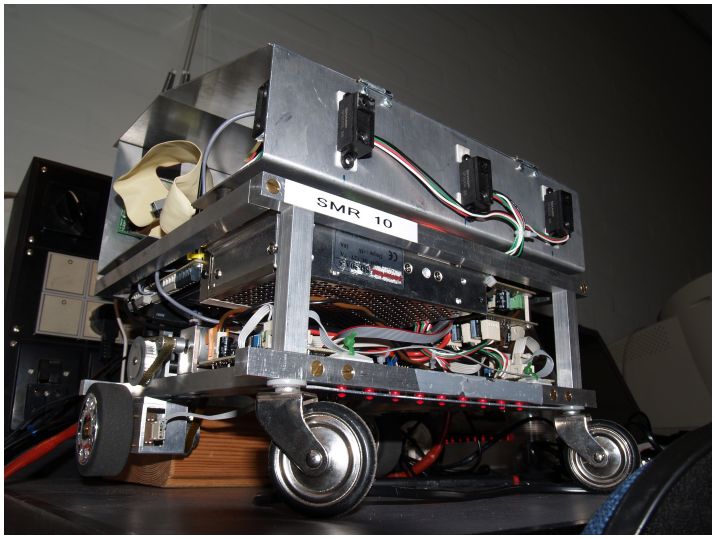

Figure 5.2: The Small Mobile Robot (SMR) platform used for the motion pattern tests.

The following example is based on encoder increments on a SMR (Small Mobile Robot) platform at the Automation and Control labs at DTU, which is an approx. 30x30x30 cm in size and differentially driven mobile robot platform, illustrated in figure 5.2. The robot was designed for teaching in mobile robotics as well as doing tests without the risks and overhead of running medium sized mobile robots. The SMR runs DTU MobotWare, which makes development

and control completely transferable to other MobotWare operated robots, and the current versions of the SMR include Hokuyo URG-04LX laserscanners and Microsoft Kinect RGB-D cameras.

In the experiment, the SMR was instructed to perform a sequence of motions, and the number of increments on the left and right wheel encoders (`encl` and `encr`), as well as the speed reference for the left and right motors (`speedl` and `speedr`) were logged at 100 Hz directly from the Robot Hardware Daemon (RHD). The motions were given in the following sequence:

Nr.	Motion	Note
1	Straight forward	Motion starts and ends with accel/deccel to 0
2	Circular arc towards left	Motion starts with accel from 0
3	Circular arc towards right	No accel/decell other than motion change
4	Straight forward	Motion ends with deccel to 0
5	Stationary turn towards left	Motion is only accel/deccel as stationary speed is not reached
6	Straight forward	Motion starts and ends with accel/deccel to 0
7	Stationary turn towards right	Motion is only accel/deccel as stationary speed is not reached
8	Straight forward	Motion starts and ends with accel/deccel to 0

Table 5.1: Motion sequence of the SMR in the experiment.

Inspection of the resulting encoder data allows manual annotation of the motion sequences, making the different motion patterns clearly visible including acceleration ramps. An interesting observation was that a number of outliers appeared in the dataset. These outliers were caused by using Python to record the dataset, which proved incapable of keeping a sample rate of 100 Hz. Missing a sample in the data synchronizations caused the encoder increments to be the double of their expected real value.

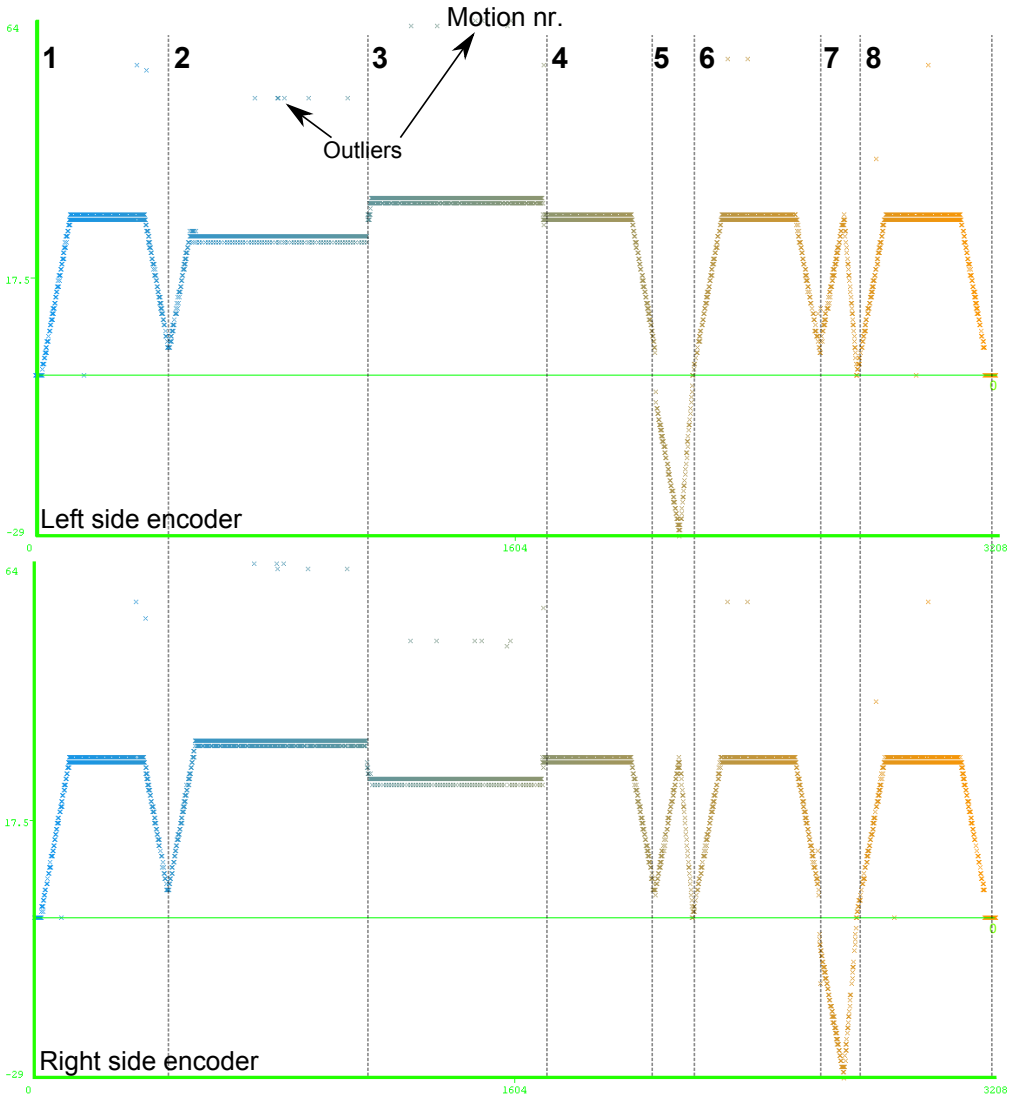

Figure 5.3: Plot of encoder increments with annotation of the motion patterns from table 5.1. The top plot is the left side encoder and the bottom plot is the right side. Also notice the outliers, which are caused by the inadequate sampling rate of the Python script used to record the data.

5.5.1 Analysis using static clustering

The first question to be asked is if the use of clustering will be capable of extracting information about the motion patterns from the data-set. All real-world sensors and estimation systems have a given degree of uncertainty. In many statistical frameworks, sensor noise is represented as Gaussian random variables. This is the basic assumption applied in many basic algorithms used in robotics, such as the Kalman filter (KF) and Extended Kalman filter (EKF). As real sensor noise often can be approximated by Gaussian distributions, the picture is different when entire data-sets are considered.

The detection challenge in situation assessment is not to estimate the real

signal beneath a Gaussian noise distribution but rather to recognize changes and patterns in sensor data over time. This data stream has the possibility of many states that might or might not be related linearly. For this representation a pure Gaussian distribution is inadequate to represent the belief state correctly, which makes it obvious to turn to the Gaussian Mixture Model (GMM) (Bishop, 2006).

Gaussian mixture distributions were defined by McLachlan and Basford (1988) as linear super positions of basic Gaussian distributions. By linear superposition it is clear that such super positions can form even quite complex distributions. Using a sufficient number of Gaussian, almost any arbitrary distribution can be approximated accurately. GMMs can efficiently be used for clustering by means of the Expectation, Maximization (EM) algorithm, where the EM algorithm assigns a probability distribution to each instance which indicates the probability of it belonging to each of the clusters. The EM algorithm would generally have been a very good choice as core clustering algorithm, but unfortunately it has a highly iterative access on the data-set and thus unsuited for clustering streaming data. In this experiment EM is applied as reference and for validating the concept of the chosen approach of data analysis.

For this purpose the data-set is analyzed using the Weka 3 Data Mining and Machine Learning software (Hall et al., 2009). Weka is developed in open source by the University of Waikato and has become the de-facto standard of open source Data Mining tools, as it contains a large number of algorithms implemented (including EM), a very good suite to analyze data and a good API for researching in new algorithms. The result after a slight parameter-fitting and more than 5 minutes of running the clustering process is shown in figure 5.4.

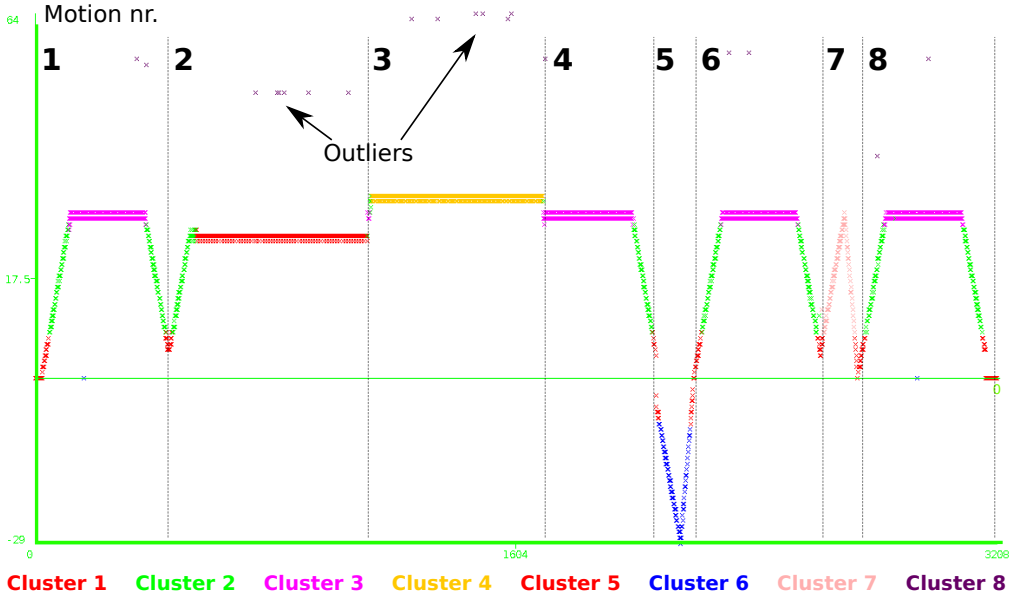

Figure 5.4: Result clustering shown for the left encoder after running the data-set through the EM clustering algorithm in Weka, where each cluster are represented by a different color in the graph.

The clustering algorithm yielded a total of 8 clusters, which naturally did not follow the motion primitives of the sequence as described in table 5.1 on page 84. The algorithm did, however, manage to capture each of the characteristic patterns of the motion sequence as shown in table 5.2. It is especially interesting that the algorithm managed to assign all outliers to cluster 8, despite that they were randomly scattered throughout the data-set.

Cluster	Motion pattern
1	Stationary or values close to 0
2	Linear accelerations with both wheels
3	Straight forward movement as in motion nr. 1,4,6 and 8.
4	Circular arc motion towards right
5	Circular arc motion towards left
6	Stationary turn towards left.
7	Stationary turn towards right.
8	Outliers

Table 5.2: Motion patterns identified by the EM algorithm.

The results in table 5.2 clearly shows that higher level information can be extracted from even simple raw sensor measurements using clustering. However, this approach only managed to segment the spatial component of the data-set, where the situational information found in the temporal structure cannot be modeled using this approach.

5.5.2 Analysis using EMM and on-line stream clustering

Using the EMM modeling framework and the on-line streaming clustering model, it is possible to model total spatio-temporal structure of the data set. Where the EM algorithm uses a mixture of Gaussian to model the cluster distributions, the proposed clustering algorithm uses a much simpler representation; the characteristics of the data-set cannot be encoded into the clusters in a similar degree. By organizing the clusters in the EMM, an even more detailed representation can be achieved, as the cluster distributions are encoded into their relations, which also contains the temporal structure.

The data set was processed using the Jaccard similarity coefficient and with a threshold of $\lambda = 0.985$ which produces a fairly high number of states but also reveals a lot of details in the structure. The resulting EMM was streamed on-line to the graph visualization tool Gephi as described in chapter 6 on page 91. Figure 5.5 on the following page shows the Gephi visualization with a manual annotation of the motion pattern structure modeled in the graph.

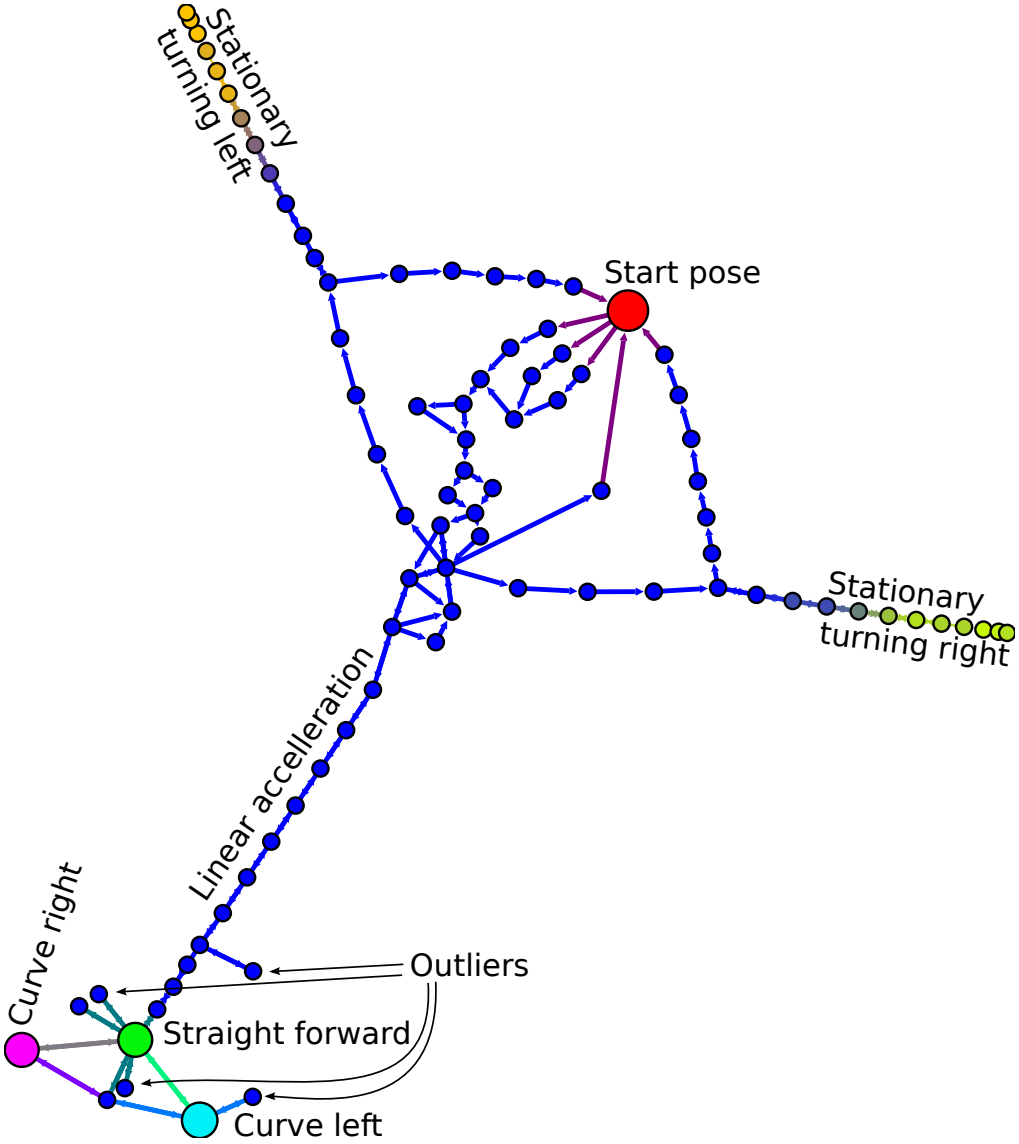

Figure 5.5: Graph visualization of the EMM model which have been learned from the data-set using on-line streaming clustering.

It is clear that the EMM model also managed to capture the different motion patterns, with stationary turns in the right part of the graph and linear together with curve motions in the left side. It also managed to capture a number of other details:

- Straight and curve motions follow the same acceleration patterns, which were also detected by the EM algorithm.
- The temporal locations of the outliers are identified. The outliers could have been deviations from normal operation caused by a wheel to be stuck, the robot touching an obstacle or a type of mechanical fault. On-line

identification makes it possible to react to these critical situations and resolve them.

When analyzing the graph around zero speed, some interesting hardware characteristics become visible. Figure 5.6 shows a close-up of the graph around the zero state, with manual annotation to identify the different acceleration and deceleration sequences. Here it is noticeable that none of the motions follow the same patterns for the initial part of acceleration as for deceleration. When speeds become higher, the curves meet and the sequence becomes identical. This can be caused by the differences in static and dynamic friction as well as electronic or software part of the motor control.

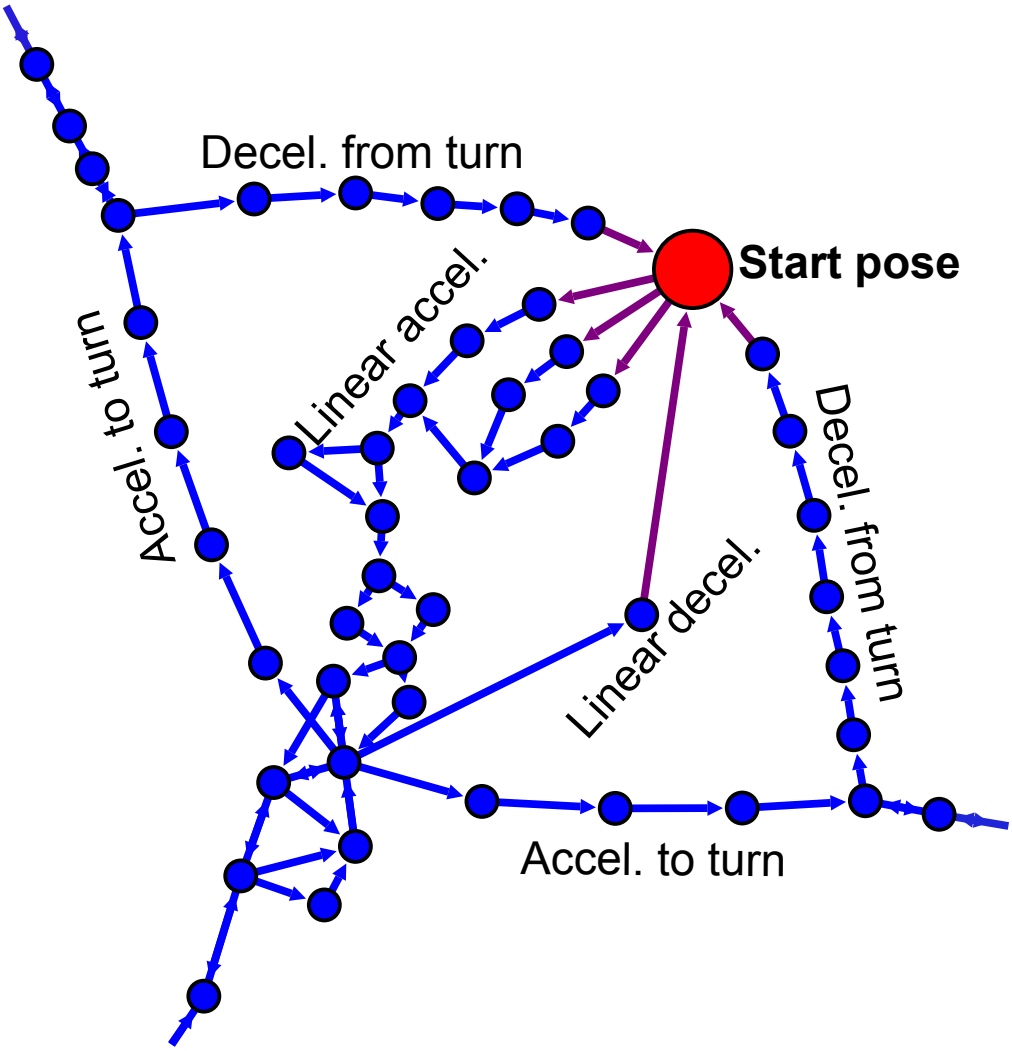

Figure 5.6: Close up of the graph around initial acceleration and deceleration to zero. Notice that acceleration and deceleration follow different patterns.

From this experiment it is possible to conclude that the use of clustering does have the capability of revealing relevant patterns of higher level information, even

from low level sensor data. Clustering using EM showed itself capable of clearly identifying the correct patterns, but at the cost of high computation time and a large number of passes through the data-set. Using EMM and the simpler nearest neighbor clustering proved capable of identifying the same patterns in the data-set using only a single pass through the data-set. Furthermore, the EMM extracted the temporal structure of the data-set, which revealed more detailed information about the outliers of the data-set and certain hardware characteristics of acceleration and deceleration.

5.6 Summary

This chapter has presented a thorough analysis of the input signal processing system for situation assessment with a focus of clustering and classification of information streams. Clustering of information streams are an efficient tool to perform on-line analysis of data where no explicit models exist and thus leaving the problem unsuitable for model-based approaches such as Kalman filters and many other statistical approached.

A clustering engine was designed to combine the strengths of both classification and clustering using the nearest neighbor principle to assign data points to existing clusters on-line or to create new clusters if no clusters could be found with an adequate level of similarity. To evaluate similarity the Jaccard similarity index was chosen as it is suitable by its range of calculation output and suitability to vectors with no explicit Euclidean relationship.

To evaluate the relevance and capabilities of using clustering and pattern recognition to identify higher level information, the effects of clustering were evaluated on data-set containing incremental values from right and left wheel encoders from a differentially driven SMR mobile robot platform during a sequence of different motions. Using the iterative EM clustering algorithm, it was possible to identify and segment each individual part of the motion sequence. Using the on-line streaming clustering engine and EMM to model the spatio-temporal structure of the data-set, it was possible to identify and segment the individual parts of motion sequence, as well as their temporal structure. Furthermore, the temporal analysis showed that process characteristics, such as friction, can be learned and identified by the model.

5.7 Further improvements

Further improvements are, however, expected to be obtainable in the design of the clustering engine by further investigation into the use of some of the emerging on-line variants of hierarchical clustering. Using a hierarchical representation may make it possible to choose the level of abstraction and detail represented in the model by selecting the appropriate hierarchical level in the model. Thus, it does require a hierarchical organization of the EMM structure as well, which could have some benefits, but also increase complexity significantly.

6

Situation Assessment Platform

The volume of software needed to run situation assessment is quite substantial, especially when the structure should perform at a level where technical details do not cloud the focus of doing experiments with situation assessment, even when working with a robot in on-line operation scenarios. In these scenarios good user interfaces and visualization together with reliable software modules are key issues. Furthermore, it is crucial that the framework architecture are as thin as possible to allow researchers to keep focus on algorithm development and testing instead of struggling with complicated data interfaces and development paradigms.

This chapter is dedicated to put together the pieces from the analysis and design of system components from the previous chapters. The synthesis will result in a full framework, designed for situation assessment integration with mobile robotic software frameworks. The presented framework, the Situation Assessment Platform (SA-Platform), provides a modular GUI-based and expandable infrastructure to develop and experiment with situation assessment, with easy to use infrastructure and visualization.

Inspiration has been drawn in particular from the modern trends of development and research within open source robotic frameworks. Special attention is put into make sure that algorithmic functionality stays separated from framework infrastructure code and that data and functional services can be accessed transparently and easily throughout the framework. Furthermore, it has been an important design criterion that core functionality can be easily extracted and later ported to a lightweight embedded version in a future on-robot version for completely autonomous situation assessment.

Figure 6.1: The Situation Assessment Platform (SA-Platform) is the GUI based modular framework for on-line situation assessment with mobile robots.

6.1 Platform architecture

A Meta model for information fusion system architectures was presented in the revised JDL model by Steinberg et al. (1999). Although this model contains information flow in both directions, there is no distinct definition on how information fusion results are connected to controllers and other information consuming modules as they exist in robotic systems architectures. For the SA-Platform the usefulness of the JDL model as practical implementation guideline is rather limited, as no JDL level 0+1 processes exist in the framework, leaving only level 2 (Situation Assessment) to be implemented in the SA-Platform.

As discussed earlier, most modern robotic frameworks, including the popular Player/Stage (Gerkey et al., 2001), ROS Quigley et al. (2009) and CARMEN (Montemerlo et al., 2003), are all architecture neutral which does not provide any encouragement to adaptation of their architecture. DTU MobotWare is the primary implementation target of the SA-Platform, which would make it obvious to implement situation assessment directly into the modular architecture of MobotWare. This will also be the solution in a future head-less and mobile robot integration oriented version, but for ongoing research it is necessary with a graphical framework, where real-time evolution of the spatio-temporal models can be followed, visualized and efficiently interacted with.

The model of Situation Awareness (SA) by Endsley (1995) describes the process of human formation of SA and has been good inspiration through the work of this thesis. Although the model is not thought as a technically implementable model, it addresses the processes involved in situation assessment in better detail than the JDL model and, furthermore, results in a principal architecture that makes particular sense for humans. From Endsley's model the architecture must hierarchically support the three levels of SA, 1) Perception of elements in the situation, 2) Comprehension of the situation and 3) Prediction of future status.

In the SA-Platform, these levels of SA are implemented in the following models:

1. **Perception of elements.**
   Information for level 1 SA is received from the robotic frameworks, either from raw data, or from Level 0+1 information fusion processes (estimation, tracking, recognition, etc.). From MobotWare, raw low level sensor data are received from the Robot Hardware Daemon (RHD) and Automation Robot Servers (AURS), where AURS also hosts Level 0+1 Information Fusion processes which produce higher level data. The heterogeneous data-source module and situation modeling framework implement the level 1 SA components.

2. **Comprehension of situation.**
   Comprehension is achieved by creating spatio-temporal relationships of level 1 SA information. This is implemented in the Clustering engine and in the EMM Spatio-temporal modeling framework.

3. **Prediction of future status.**
   Prediction of future status is achieved by calculating the maximum likelihood of states traversing from the current node in the EMM model to other nodes, which are critical parts of given situations. Level 3 SA is implemented in the prediction engine.

6.1.1 Design decisions

Besides the architectural structure, practical research development in a mobile robot context leads to a number of requirements, which set good development practice and the necessary supporting functionality for efficient research. The following list summarizes the key framework requirements.

Data sources must be heterogeneous.
It should be possible to process multiple information sources from any part of a mobile robot system, at multiple levels of abstraction and at multiple data rates. Furthermore, the expansion towards new sources of data should be simple and only require limited insight to the core system integration details.

Multiple situation assessment instances must be able to run in parallel.
Each assessment instance can contribute with minor clues to assist robot control, major clues about the status of complete robot functionality or even provide information to higher-level situation assessment instances. Hence, running several instances in parallel enables tracking of mulitiple situations simultaneously.

Visual representation of the produced graphs in real-time.
Manual and on-line inspection and debugging of temporal systems can be challenging, as patterns can be cycled through many times and information is encoded in the order of these sequences. Therefore, on-line visualization is a key tool to follow the system behavior through execution.

Separation of functionality and framework.
The framework must support and encourage a clear separation between exclusive function modules and framework components, such that function modules can be expanded and modified without modification of the stable framework core modules. Furthermore, the separation into function modules makes the exportation of modules e.g. to a simpler embedded head-less framework possible.

Combining the structure from the SA model by Endsley (1995) and the design requirements leads to the architectural structure of the Situation Assessment Platform, illustrated in figure 6.2, which also illustrates the information flow of the architecture.

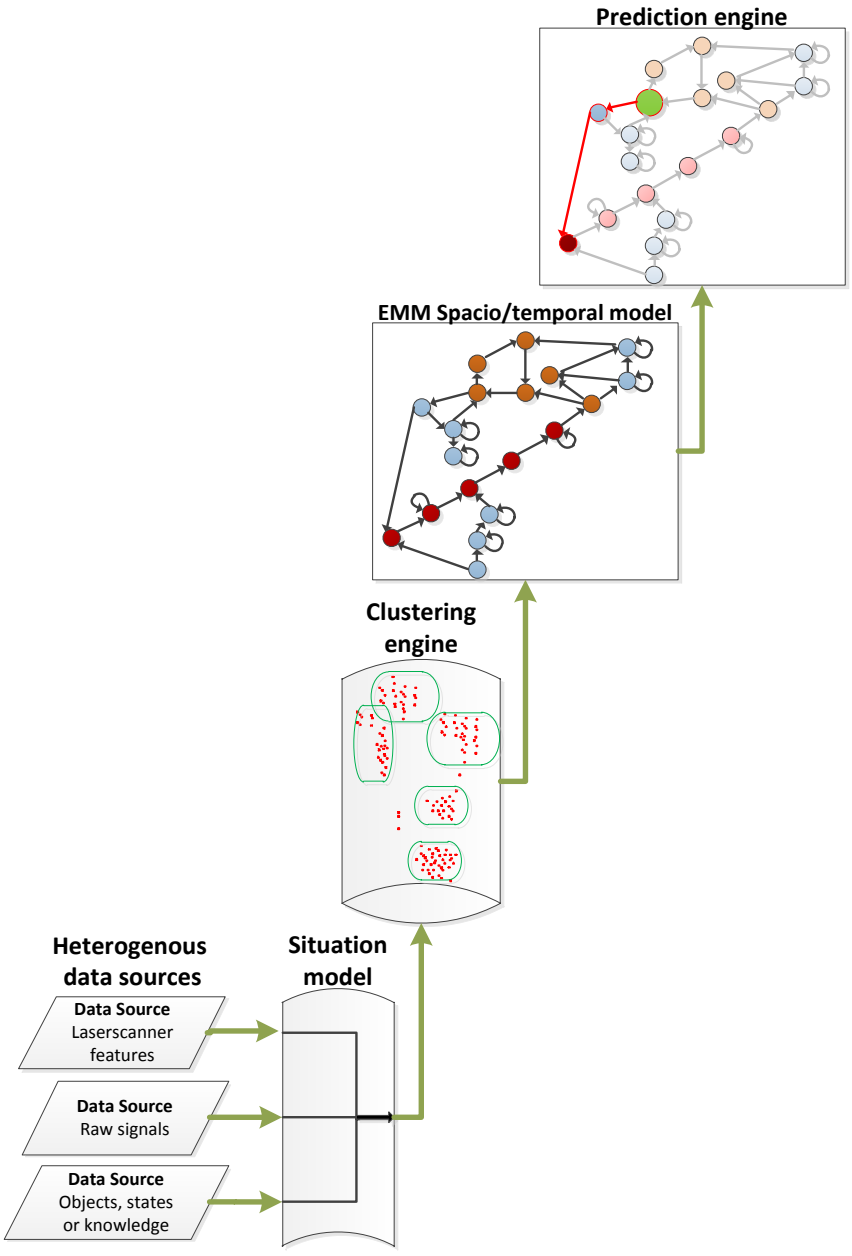

Figure 6.2: The basic elements of the proposed situation assessment architecture. Information from many data sources can be fused by the situation model. The clustering engine groups and matches the data in clusters, which are structured in the spatio-temporal domain by the EMM model.

The situation model (described in section 4.4.1 on page 70) defines which sources of data that should be part of the situation perception and fed into the clustering engine. The clustering engine matches and creates cluster states from

the arriving data and feeds it into the EMM Spatio-temporal model. The EMM model builds a Markov Chain of states from the sequence of clusters, tracks the current states and updates transition likelihoods between states in the graph. Finally, the prediction engine continuously calculates the maximum likelihood of transition from the current state to other states in the Markov Chain in order to predict the likelihood of transitioning through critical sequences.

This architecture contains many modules which receive and process information simultaneously. In order to monitor the many concurrent processes and at the same time follow the temporal development of the EMM model, it is necessary to use a graphical user-interface where many functions can be illustrated, manipulated and monitored at the same time. For this purpose the Situation Assessment Platform is implemented building on the Java NetBeans Platform[14], which contains an enormous volume of easily implementable features for building a rich graphical user interface and yet the platform is built upon a clear-cut modular system which helps to keep functionality in separate library modules with a limited number of dependencies. The modular system is very beneficial to ensure that UI and functionality do not get too intertwined and allows a future port to a headless robot-system, which does not require direct on-line monitoring.

6.2 Heterogeneous data sources

Being a higher level information fusion process, it is necessary to support data input from a large range of information sources. To facilitate this the Situation Assessment Platform has a plug-in based data-source back-end. Using the ServiceLoader architecture in Java, data sources are loaded as independent modules, where data consumer modules can subscribe to their information at runtime without any source code coupling or compilation dependencies.

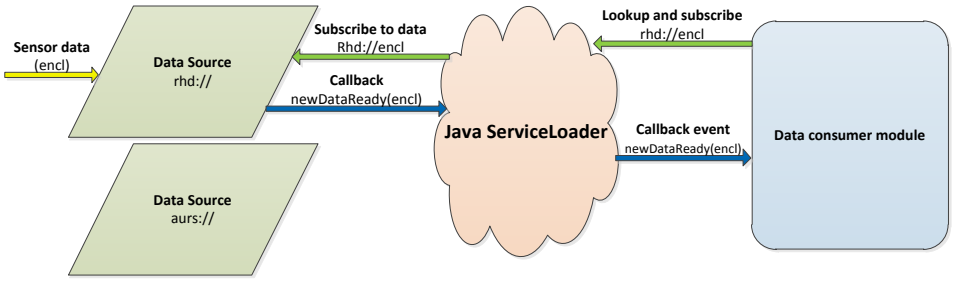

Figure 6.3: Flow of the data source service architecture. A data consumer module subscribes to the RHD data source through the Java ServiceLoader API and receives a callback each time new sensor data are ready.

Figure 6.3 illustrates an example of a data consumer module, who wants

[14]The NetBeans Platform, `http://netbeans.org/features/platform/`, 2012, Oracle Cooperation

to subscribe to the RHD data-source to receive the left encoder values: `encl`. The module initially makes a lookup in the Java ServiceProvider API to find the `rhd://` and subscribes to the `encl` variable. If the lookup is successful, the RHD data-source module registers a callback to the data-consumer. When new values are received from RHD, a callback is invoked to the data consumer module (and other possible subscribers) which then can process the new data. Using this method, the data-source modules only need to retrieve the subscribed data from their sources which greatly reduce bandwidth and resources.

All parts of the data source service interface are designed using interface classes, which make it generic and efficient to create new data source modules. A new module is created by wrapping existing interface libraries or writing interfaces to new data sources and then just implement a few abstract functions to insert the source module in the Situation Assessment Platform. At the current time of writing, the Situation Assessment Platform contains the following data source modules.

- Native client interface to Robot Hardware Daemon supporting data rates in excess of 100 Hz.
- XML-based interface to AURS with publish-subscribe of all data-variables.
- Socket interface to a dedicated ROS-node with a publish-subscribe connection to all topics in a ROS system.

6.3 Situation modelling

The situation model can be considered the recipe of elements to perceive in order to comprehend a situation. It can be compared to the specific instruments a fighter pilot must monitor in order to predict if the aircraft is going to stall during a steep climb or the indicators a power plant operator monitors in order to comprehend situations such as leakages or pump errors.

The situation models are based on a XML configuration file, which defines each situation instance with the following parameters:

- Clustering algorithm and threshold
- Data sources and associated variables and selected indices
- Predefined states (e.g. well-known critical states defined by an expert)
- Loading of previously learned EMM model in binary form

Situation models can be used to track and detect information at many levels of abstraction, for example simple low level events as the correlation between wheel encoders and commanded wheel velocities as investigated in section 5.5 on page 83 to discover anomalies like a wheel being stuck, mechanical faults or increased vehicle loads (intentionally or unintentionally). Situations could also

be of higher level, such as the correlation between states in a mission sequence and the map pose of the robot.

An example situation model for tracking the relationship between EKF localization covariance (diagonal elements 0, 4 and 8 in an array format) and the (x, y) map coordinates of an AGV platform can be used to detect if the localization system is having abnormally high localization uncertainty at given locations (further investigated in section 7.3 on page 128). Using the localization and mapping modules of AURS, the resulting situation model is shown in listing 6.1.

```
<emm name="AGV State tracker" >
 <model name="CovarTrack">
  <clustering method="Jaccard" threshold="0.8"/>
  <probability enable="true"/>
  <data source="aurs" variable="localize.covar" index="0 4 8"/>
  <data source="aurs" variable="mappose.pose" sync="true" index="0
      1"/>
  <load model="covar.bin"/>
  <state name="Initial" data="0.5 0.5 0.5 0.0 0.0" />
 </model>
</emm>
```

Listing 6.1: Example of XML configuration of a situation model to monitor the relationship between localizer covariance and map pose for an AGV platform.

Data sources have many different rates of data, often with raw sensors running up to hundreds of Hz and others only run at a few Hz. Often, there is no meaning in updating the relationship between the data at the change of all variables, but synchronization should be tied to the updates of specific variables. In the example in listing 6.1, the model is updated on synchronization to the `mappose.pose` variable, which is the last of the two variables to be updated. The `probability` parameter enables the prediction engine which calculates the maximum likelihood for transition from the current state to any state in the graph at each update.

6.3.1 Loading of Models

The spatio-temporal EMM models can be learned from scratch from sensor data, but often previously learned models are necessary to perform situation assessment already at system start-up. Two methods are supported to pre-load EMM models, where the first method is hand-crafting the states in the situation model XML file, using the `<state/>` identifier. This is especially useful for modeling by experts, who want to ensure that center-points of data-clusters are placed correctly e.g. known end-positions for position tracking through a motion sequence. The second method is to load a binary EMM model, which has been saved after earlier runs. Figure 6.4 on the facing page illustrates the loading and creation process of the EMM.

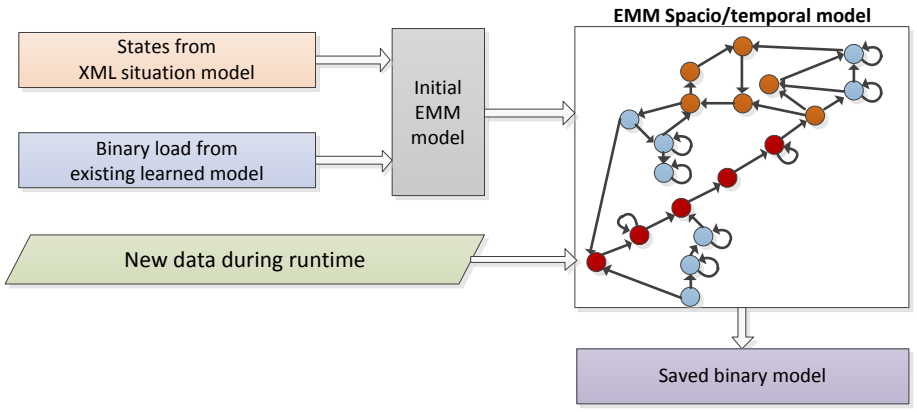

Figure 6.4: Pre-defined structure of EMMs can be loaded in several ways. States can both be loaded from XML situation models and from binary versions of models which have been learned through earlier sessions.

The EMM loading system also supports the combined loading of states in binary form and from the situation model. In this case, the binary model is loaded first, then followed by states from the situation model. Before the states are loaded from the situation model, the binary model is searched for matching states, which then are merged with situation model states if they are found.

6.4 Pre-processing of data

Sensor information is not always in good format for higher level fusion. The original JDL Data fusion model by White (1987) explicitly defined a stage of source pre-processing, as often it is necessary to do simple re-formatting of the original data to suit higher level fusion processes. The INFERD Fusion Engine (Sudit et al., 2007) uses an Input Manager to transform raw input from cyber-attack sensors into a unified format suitable for situation assessment, but no calculations or feature extraction are involved in the process. Duda et al. (2012) proposes the use of equi-width cubes for data-reduction in the pre-processing stage to limit noise and deal with too high data rates. In robotic systems pre-processing is often the process of extracting important features or composition of measurement from bulk raw sensor measurements.

The Situation Assessment Platform does not explicitly have a pre-processing module built into the platform. Within DTU MobotWare lower level fusion processes are implemented in the AURS framework which is the most obvious data source for situation assessment. Besides fusion and perception algorithms AURS also contains an interpreter module for a rule-based scripting language (Andersen et al., 2010). The rule interpreter was specifically designed as the executor of an easy and efficient scripting language to orchestrate the functionality of AURS. Andersen et al. (2010) presents how the rule-bases scripting language can be used elegantly as mission management tool for autonomous tractor op-

erations in an orchard and where the concurrent interpretation of rules proves efficient to coordinate auxiliary tasks, such as stopping insecticide sprayers at the ends of tree-rows as a script which can operate independently without being mixed into one main script.

Besides being useful for monitoring execution status and scripting mission sequences, the rule interpreter is also suitable as data pre-processor. The rule-based system supports basic algebraic operations (from `math.h`) as well as a number of robotic-suitable tools such as pose (x, y, θ) coordinate system transformations. Using these operations sensor data streams and fusion results can be processed e.g. to extract basic features from laser scans, transform relative poses between objects or to calculate Δ-values from continuous measurements. Rules are executed at 4 Hz in AURS with default settings, which is sufficient for most situation monitoring. An example of pre-processing rule is illustrated in listing 6.2.

```
<rule name="dirDetector" run="true">
 <init>
  global.dir.run = true #Run the rule
  <rule name="dirdetect" if="(global.dir.run)" >
    mPose = mappose.pose        #Extract the map pose
    mAng = mPose[2] * 180 / Pi #Convert pose to degrees
    #Calculate the closest direction
    global.dir.dir = int(mPose / 22.5)
    print "Direction detected: " global.dir.dir
  </rule>
  print("------------ Direction tracker initialized")
  </init>
  wait() : false
</rule>
```

Listing 6.2: Pre-processing rule used to reduce the robot orientation to a principal direction indication.

The preprocessor in listing 6.2 extracts the orientation from the robot mappose and calculates a principal direction indication by dividing the orientation by 22.5°. This direction indication can be applied to investigate if the robot follows the same pattern of orientation in a given motion without having to deal width minor fluctuations in orientation.

For more complicated feature extraction and calculations which exceed the rule-based scripting language, AURS has a well-established plug-in system where feature extraction algorithms can be implemented efficiently in C++. Within the plug-in system there is similar easy access to all resources of AURS, thus with the overhead of preparing a source-code module with classes, headers and build-system adjustment.

6.5 Component structure

A modular software structure has been the de-facto development standard within robotics since the community driven frameworks became popular in the late 1990s and early 2000s. The key goal of developing robotics software in modules is to ensure that the software can be re-used for other purposes without being intertwined, and thus leading to global re-use of software components written by experts in their fields. In modular software development, two layers of modularity exist.

The first layer of modularity is to use a modular software framework (e.g. Player/Stage, ROS, MobotWare, etc.) as implementation platform. These platforms encapsulate functionality in modules which ensure well defined APIs and dependencies such that the modules easily can be transferred to the same framework in other systems or contexts.

The second level of modularity is within the source-code itself. A common caveat is to mix drivers and algorithms into framework specific code, which ties the functionality of modules to the frameworks and often makes it difficult for others than the author to work on the modules to improve the core algorithms without being a framework expert. Makarenko et al. (2007) analyses the

structure of framework specific code vs. drivers and algorithms in a number of popular robotics frameworks. Only one (Orocos) out of six frameworks has managed somewhat separation between drivers and algorithms and the robotic system framework. Others mix the code which severely limits the portability of the code. What is achieved in modern robotic frameworks are by Makarenko et al. (2007) called *the marketplace*, which in many aspects has similarities to the app-markets known from Smartphones. Modules for a specific framework can be shared and applied quickly, but transfer to other platforms are at the same complexity level as a complete rewrite.

The structural design goal of the SA-Platform is to ensure that even though it is developed using a graphical front-end and the sophisticated NetBeans platform, it must be possible to re-factor the core modules into a lightweight version, suitable for integration into embedded systems. This could either be into a MobotWare AURS plug-in or as a stand-alone application. To enable this, the core of the architecture is based on five function libraries, which in principle could be integrated into one tightly coupled application as illustrated in the architecture overview in figure 6.2 on page 95. The modules are:

- The DataSource library
- The XML Situation model loader library
- The Clustering engine
- The EMM Modeling engine
- The Prediction engine

In order to integrate the libraries into the Java framework without causing any source code dependencies, the Java ServiceLoader is used, not only for the DataSourceService as described in section 6.2 on page 96, but also to publish updates of the EMMs to subscribed modules. The design choice of using a second ServiceLoader interface at the EMM model is based on a hierarchically layered structure of the software components in the SA-Platform.

6.5.1 Hierarchical structure

A hierarchical structure of software components is a natural and common way of organizing information processing of increasing levels of abstraction. However, as discussed earlier, a hierarchical structure which enforces a certain organization of modules, interfaces and decision structure has a theoretical appeal and benefits development in large and complex systems, but in practice it is hard to implement and adapt to the requirements of tackling real-world problems. I believe such a structure can work well when it is used by the original inventor and maybe also by collaborators at the same department with a similar understanding and mindset, but history has proven it is difficult to apply strict architectures in larger open source communities.

Similar problems led to the revision of the JDL Data Fusion model. The original model (illustrated in figure 4.1 on page 55) proved it difficult to implement,

as it easily became interpreted as a canonical guideline for an entire software framework, where the model dictated which level of abstraction was processed at each level. The revised model has a lot of similarity with a traditional Sense-Plan-Act architecture, and like the NASA NASREM architecture (Albus et al., 1989) the model defines that all layers can process information from level of abstraction, but the process types and resulting information abstraction level are fixed towards the architecture model. As a principal sketch for understanding the roles of system components, I find the model beneficial, but when dealing with practical implementation and real-world systems it can be too restricted to limit the roles of system components to a certain type of processing and output.

The component structure of the SA-Platform has been inspired by the architecture implemented by Diankov and Kuffner (2008); Diankov (2010) in the OpenRAVE project. OpenRAVE is a software architecture designed for control and motion planning for humanoid robots. The versatile nature of humanoid robots pushes the architecture away from the navigation-centric paradigm which has dominated traditional architectures for mobile robots. The principles of the OpenRAVE are an architectural breakdown by the functional services each module provides within the architecture and with API interfaces similar to ROS. Where a node-interface in ROS is based on the type of information which is transferred across the interface, the interface in OpenRAVE is based on module services, such as Planners, Controllers, Sensors, Robots, etc.

In similar fashion Java ServiceLoader service interfaces in the SA-Platform are used to expose certain services within the framework, based on their functionality.

Another source of inspiration to the system architecture is the three levels of situation awareness described by Endsley (1995) where the interfaces implemented represent one conceptual layer of situation awareness:

1. **Perception of elements** is exposed through the DataSourceService.
2. **Comprehension of situation** is exposed through the EMMService.
3. **Prediction of future status** is implemented in the prediction engine.

Both SA level 1 (perception) and level 2 (comprehension) are built around the Java ServiceLoader to make these interfaces available in the entire framework, and yet keep them free of tight independence couplings. The situation model loader bridges these layers as it creates the DataSourceService connections between data sources and clustering engine.

The clustering engine and the EMM model work in tight connection, clustering sets of perception data and organizing it in the EMM model. The EMM model is published through the `EMMService` interface, which issues callbacks to all subscribers when the EMM model is updated. This way, the GUI is updated with the changes in EMM structure, updates are streamed to the visualization and the prediction engine updates transition probabilities.

Expanding the basic system architecture from figure 6.2 on page 95 to the hierarchical structure, the resulting system architecture is illustrated in figure 6.5.

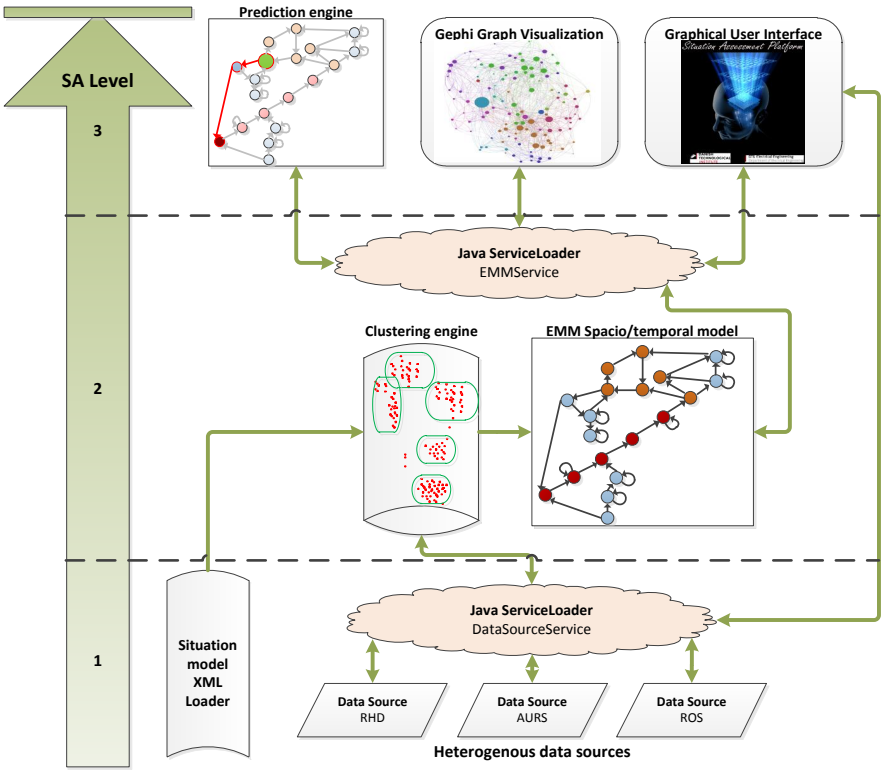

Figure 6.5: Structure of components in the SA-Platform. The platform architecture is separated hierarchically in three levels, using the ServiceLoader to define service interfaces at each conceptual layer of situation awareness.

At SA level 3, the visualization and GUI components are also illustrated, as their role is considered as a part of involving the operator in both the comprehension and especially the prediction of future status.

6.6 On-line data visualization and inspection

Markov chains are very illustrative for humans when they are visualized as directed graphs. A directed graph shows the spatio-temporal properties of the model as well as general structure. When observing graphs, the structure of even fairly high volumes of information becomes easily interpretable. A common example is computer file systems which often are illustrated as trees (a special type of graph) where thousands of files can be navigated without much effort. Directed graphs are often visually used to illustrate road maps. Within robotics, graphs are commonly used for visualization and planning of navigation

routes.

When directed graphs become relatively large, they become fairly difficult to visualize and interpret. In the case when graphs have sizes like common road-maps, it is possible for any human interpreter to investigate the structure to extract details of the graph. When they grow beyond a certain size, the individual elements of the graph become hard to distinguish and information from the graph must be identified using analysis tools and algorithms (Herman et al., 2000). Figure 6.6 shows an example by a graph rendering of my personal connections from the social network Facebook [15] and their individual connections using the graph visualization and analysis tool Gephi (Bastian et al., 2009). When loading the raw graph it is a big mess, but applying a basic layout algorithm and a color-coding based on node modality (a clustering into clusters with high degrees of mutual connections) yields a graph visualization in figure 6.6 which is very useful for data analysis.

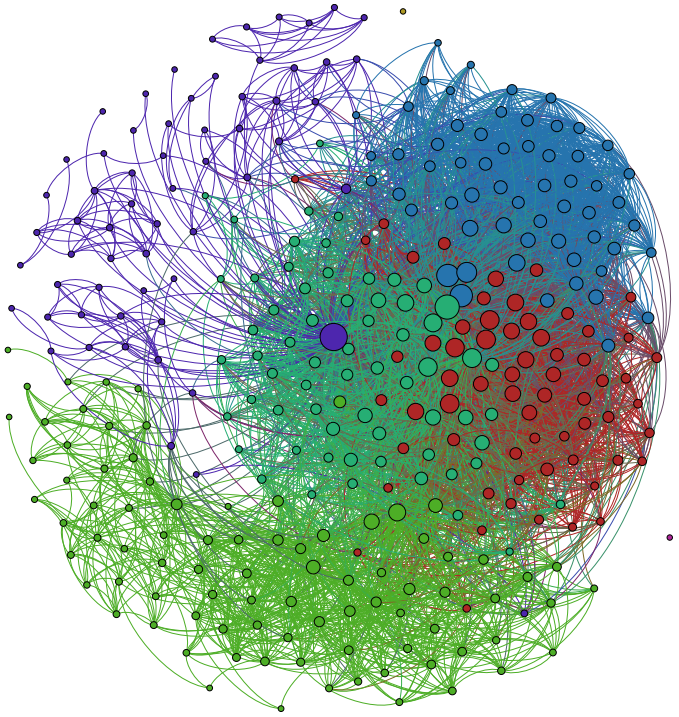

Figure 6.6: Graph visualization of a large network with many edges. This example is the first order connections and their individual connections of my personal social network on Facebook. A graph of this volume is impossible to analyze in raw form, but when layout and partition algorithms are applied, higher level clusters become evident and lead to better understanding of the complex graph structures. In this example the color coding actually does represent different social groups of my connection, all automatically extracted by algorithms with default settings using the Graph visualization and analysis tool Gephi.

In the simple example of post-processing of the graph, details emerge which previously drowned in the vast volumes of nodes and edges. In the example

[15] Facebook, www.facebook.com, 2012

in figure 6.6, groupings in my social network become evident, which even was identified correctly just using basic algorithm parameters. When the graphs are well-structured and visualized, even complex structures can be analyzed, first from the high-level structure which makes it possible to identify points of interest for closer analysis.

6.6.1 Graph visualization in the SA-Platform

Visualization of the graph-structures in the SA-Platform is done like the example by using the 3D graph visualization and analysis tool Gephi (Bastian et al., 2009) which is designed as a toolkit for visualizing and exploring large graphs. Gephi uses a fast OpenGL accelerated 3D rendering engine, which allows on-line rendering and analysis of the graph structure.

The module for graph streaming in Gephi enables on-line streaming of data to the graph workspace from external sources, either by letting Gephi pull from a source server or by pushing into Gephi in server mode. The `EMMStream` module of the SA-Platform pushes updates of the EMM graph structure to Gephi during on-line clustering in JSON message format. The server interface provides full control of Gephi to add, remove and change the graph structure.

The Gephi is ideal to visualize the temporal behavior of the graph structure during real-time operation, as the graph can be updated continuously and properties of the graph can be visualized using color coding. This makes it significantly easier to overview complex structures and properties, compared to GUI printouts of numeric values. The on-line visualization is implemented using the following properties:

- Highlighting of current node in red color and large size
- Last visited nodes visualized with a fading color trail
- Likelihood of next nodes in green color code and thicker edges to predict transitions
- Possibility to visualize maximum likelihood routes between current node and selected critical nodes

Figure 6.7 illustrates an example of the on-line visualization of a simple graph example.

Figure 6.7: Visualization example of an EMM streamed to Gephi. The current node 4 is highlighted in red and larger in size. Previously visited nodes fade back to blue after 3 transitions and states for subsequent transitions gets highlighted in shades of green based on the transition likelihood together with the weight of the connecting edges.

Besides the on-line tracking of EMM development and clustering, all other EMM structure information is also streamed to Gephi for examining the graph structures using the graph analyze tools within Gephi. These tools are particularly suitable for analyzing an EMM model, which has been generated e.g. by performing a number of docking operations. By off-line analysis critical situations can be categorized, and the model can be examined to reveal possible undesired behavior. With these annotations, the EMM can be stored and then re-loaded as prediction model for later docking operations.

Like the SA-Platform, Gephi is built upon the NetBeans platform which creates a common look and feel. Furthermore, it also exists as a stand-alone tool-kit library for integration into other applications on the NetBeans platform, but unfortunately only back-end part of Gephi is included in this tool-kit and not the real-time graph renderer, which is the most important part to use with the SA-Platform. The excellent UI and tools for graph analysis are another strong benefit of using Gephi as stand-alone application, which in total leads to the conclusion that effort is best spent by keeping the two as separate applications.

6.7 Summary

A software platform is proposed for encapsulating the modules created for situation assessment for mobile robots. Building upon the Java NetBeans platform, it features a rich GUI environment with a modular structure which eliminates inter-module dependencies through the use of the Java ServiceLoader services. This way, modules can be removed and added with a click of a button and both GUI components and internal functional modules accept the new service providers, which only exist when module plug-ins are loaded into the platform.

The role of components within the SA-Platform has been designed to follow the model of human situation awareness by Endsley (1995). This ensures the mindset of a situation assessment process and connects well to the mental model of the human who will design the situation models required to form the situation perception layer.

The overall architecture emphasizes software modality, reuse of components and ensures that transfer to embedded applications is possible. Inspired from best practices of modern open source robotic frameworks, there is no enforced

architecture within the SA-Platform, but modules can interconnect freely between each other using Java ServiceLoader services. Any type of data source can be integrated using the DataSourceService interfaces, which enables quick adaptation to new robotic frameworks or direct connections to smart sensors.

Data pre-processing is not an integral part of the SA-Platform, but must be implemented at the data source. MobotWare AURS provides an excellent infrastructure for this, either by the rule-based scripting language for low dimensional and low rate calculations, or through C++ based plug-ins for fast processing of higher data volumes and data rates.

On-line visualization is implemented using the Gephi graph visualization and analysis tool. This significantly enhances the capacity of the observer to follow sequences through fairly complex graphs. Furthermore, the analytical capabilities of Gephi are of great benefit for in-depth analysis of larger graph structures.

7

Experimental Evaluation

Situation assessment can be applied to a myriad of different purposes and applications. The purpose of this experimental chapter is to use a number of experiments to investigate the performance and usability of the proposed framework for situation assessment. In the chapter the design of situation models is specifically addressed together with in-depth analysis of the achieved models and their capabilities within perception of elements within the situation, comprehension of current status and prediction of future events.

The first experiment investigates a situation assessment module to monitor the environment of a mobile robot to detect if the robot is approaching narrow passages which might require a change in navigation strategy in order to proceed safely. The module is independent of the actual robot motion, but uses a feature extraction module to pre-process raw data from a laserscanner and hereby detect the signature of an approaching narrow passage. From the assessment model, both the comprehension of the current status of the robot in relation to the narrow passage and the prediction likelihood of future status can be extracted.

The second and third experiment investigate an AGV logistics robot application demonstration, which has been built in parallel at both DTI and DTU. In the experimental scenario, the mobile AGV platform transports a cart between two locations in a continuous loop with a small route between the transports. The demonstration is prone to have localization errors, where the second experiment is specifically designed towards detect this error as deviations from known behavior. The third experiment investigates a large data-set of cart transport. A comprehensive situation model is designed to capture the behavioral signature of the robot and its control while performing the application tasks in order to form situation awareness by comprehension and prediction in learned structures of events.

Situation assessment faces the challenge of performing processing and clustering in a large volume of high rate data. In order to tackle this challenge using hardware suitable for mobile robots, the last section of this chapter investigates the use of acceleration in stream-based clustering using mobile graphics processing units (GPUs).

7.1 Narrow passage detection

A classic critical situation for mobile robots is the move through narrow passages such as doorways or between obstacles. Especially for medium sized robots (50 cm to 80 cm wide) passage through doorways is often difficult and require constrained control strategies.

Mobile Robot navigation through open spaces is often done by route planning between two points in the global map and using the same global position coordinates for control along the path. This global navigation scheme introduces some amount of uncertainty, as it can be difficult to map an entire building with adequate precision to position the mobile robot in e.g. a doorway with only 5-10 cm of space at each side. Furthermore, the global navigation planners rarely ensure that the mobile robots have the correct orientation when they arrive at certain points, such as a doorway. An example is found in the description of the ROS navigation stack, where it is explicitly remarked: *"it may have difficulty with large rectangular robots in narrow spaces like doorways"*[16].

To navigate safely through a doorway, specially crafted algorithms should be used, which calculates path of the robot relatively to the actual local measurements of the doorway opening to make sure that the robot approaches the doorway in a suitable angle and keeps at the center line of the doorway.

Such narrow passages could appear in well-known and predicable positions, such as doorways in mapped environments, but they could also appear unexpectedly when obstacles are placed in the operating environment, such as pallets, boxes or tools, creating narrow passages in the previously open spaces. In such scenarios, automatic detection and prediction of this critical situation are very useful for avoiding navigation errors and collisions.

7.1.1 Situation analysis

The situation assessment strategy addressed in this experiment is the Type 1: detection and prediction of events in known sequences. In this experiment, sequences are learned though a number of passages through doorways which are used for on-line classification of situation events and prediction of future events.

First step is to create a level 1 situation model which is capable of capturing the characteristics of the mobile robot approaching a narrow passage and the passage through it. The mobile robot can approach a passage in a number of ways as illustrated in figure 7.1. The approach could be perfect straight-on and centered, but it could also be offset towards one of the sides or approaching at an angle. It is important to select parameters so that the variations of approach to the narrow passage are all covered within the same temporal sequence to make a common classification and prediction of when the robot approaches the passage, no matter if it is offset or rotated.

[16] ROS Navigation stack summary, Eitan Marder-Eppstein, http://www.ros.org/wiki/navigation, 2012

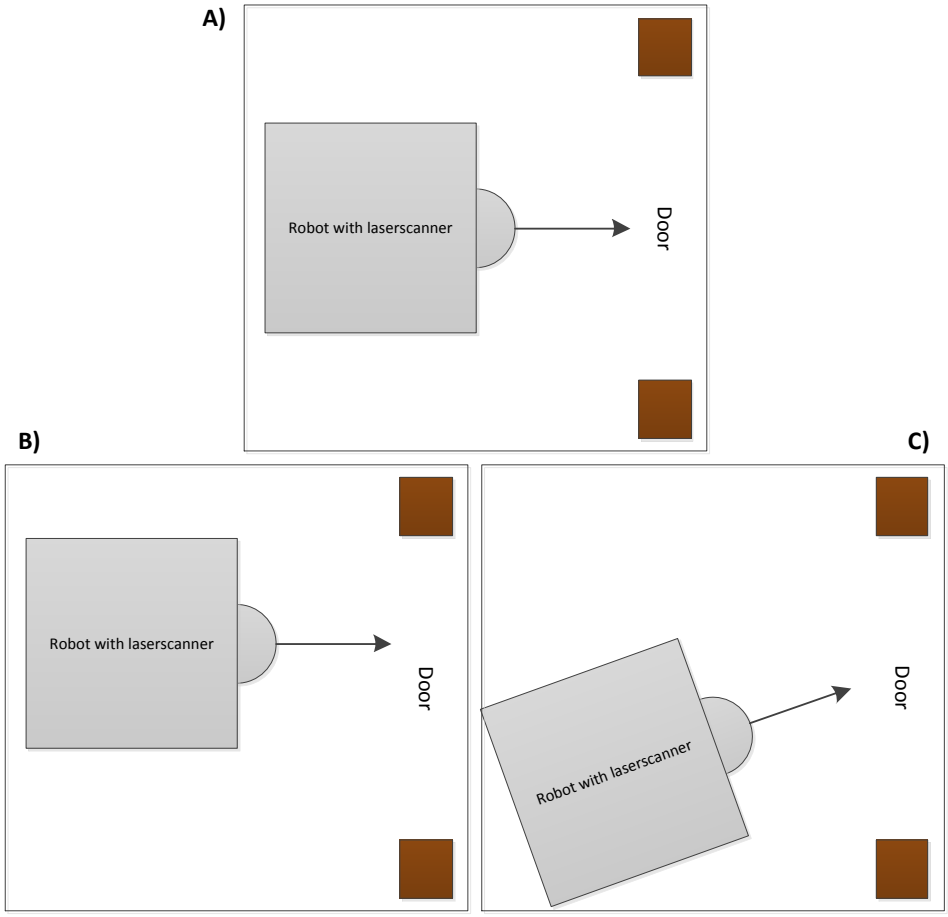

Figure 7.1: Movement through narrow passages can either be straight centered through the passage as in scenario A) or either slightly offset towards one of the sides as in scenario B) or passing through at an rotated angle as in scenario C).

A key sensor for detection of the natural environment contours and obstacles is the laserscanner. The laserscanner is used on almost all mobile robot platforms for research purposes for navigation (few newer platforms uses cameras exclusively). Also almost all commercial AGV systems use them for safety (collision avoidance) and some for navigation as well. That makes the laserscanner an obvious choice as sensor source for narrow passage detection. The laserscanner collects a range of distance measurements of 180 to 270 degrees with an equal spacing between each measurement, often 1, 0.5 or 0.25 degrees. Within such a scan, the measurements from the laserscanner will capture a complete contour of the environment within range, as illustrated on figure 7.2.

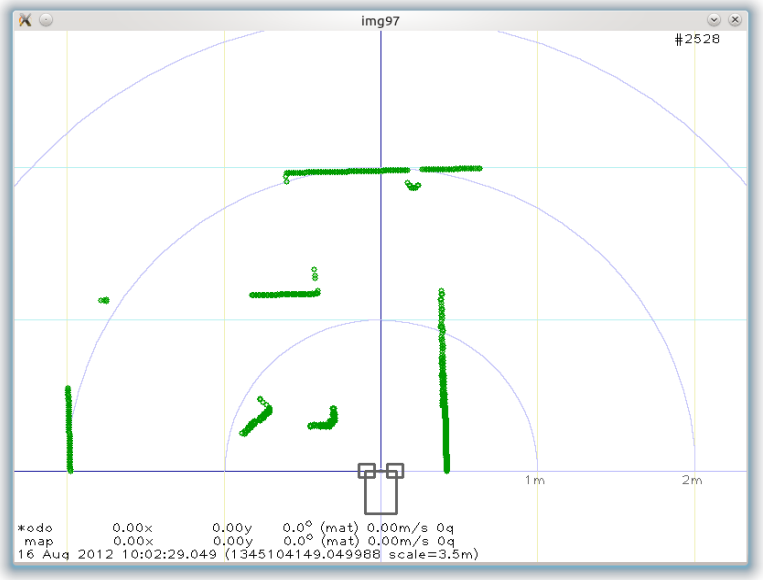

Figure 7.2: Laserscan of environment contour, looking through the doorway at a common office in DTU Building 326.

If the robot always would approach the same doorway centered and at the same angle, all points of a complete laserscan could be fed directly into the situation model. That would result in an EMM model being generated from a certain sequence of enormous data-sets of 180 to 1080 values. This approach would be extremely sensitive to any change in the environment, change in the robot approach angle or offset or just by the noise generated when the same obstacles are measured from slightly different positions (caused by the update frequency of the scanner). So, besides being heavy load for the clustering engine, using complete scans are unlikely to give robust detection of the desired signatures in the environment.

To pre-process the high volume laserscanner data, a pre-processing plug-in for MobotWare AURS was designed and used to extract features from the laserscan. The hypothesis of the feature extractor is, that the robot, when approaching a narrow passage, the closest points on either side of the robot will represent the inner corners of the obstacles which form the narrow passage as illustrated in figure 7.3.

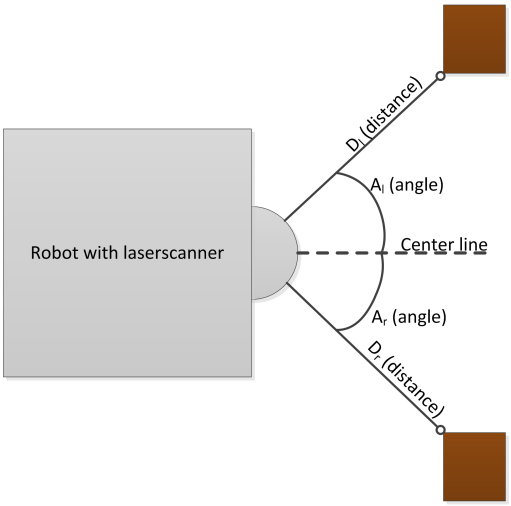

Figure 7.3: The laserscan is separated into two halves, where the distance and angle to the closest point in each side are extracted as features.

The laserscan L, containing n_l points in polar coordinates, where each point p_{scan} is represented by a distance and the angle of measurement $p_{scan} = (r, \theta)$ with relation to the center line of the robot. To characterize the closest point at either side of the robot p_l and p_r, the laserscan is divided by two halves along the center line of the robot and the points are found by minimization.

$$p_l = \min_{x \in [0; \frac{n_l}{2}]} L(x) \qquad \text{Left feature point} \tag{7.1}$$

$$p_r = \min_{x \in [\frac{n_l}{2}; n_l]} L(x) \qquad \text{Right feature point} \tag{7.2}$$

Two features are extracted from each feature point, where D_l and D_r are distances to the closest point on left and right side of the robot center line. A_l and A_r are the angles between the D_l and D_r vectors and the center line.

$$D_l = |p_l| \qquad A_l = \arg(p_l) \tag{7.3}$$

$$D_r = |p_r| \qquad A_l = \arg(p_r) \tag{7.4}$$

$$\tag{7.5}$$

These four features are able to isolate the signature of an approach towards a narrow passage from the remaining points of a laserscan. But they are not significantly invariant to differences in angle of approach, as well as offset from center of the passage. Differences in angle will change the relation between A_l and A_r and offsets will especially be visible in the relation between D_l and D_r. To compensate for this, the feature data-set s_n was calculated containing the two variables d, which are the sum of the two distances, and a which is the difference between the angles, as defined in equation 7.6.

$$s_n = \begin{cases} d = D_l + D_r & \text{Sum of distances} \\ a = A_l - A_r & \text{Difference of angles} \end{cases} \tag{7.6}$$

The d feature is resistant towards offset of the robot from the center, as any offset to one side will just transfer distance towards the other. The a feature describes the "opening" angle between the two points, which are invariant of any rotation of the robot. Tests have shown this feature set to be sufficient to give a robust recognition of narrow space passage.

7.1.2 Door passage using simulated data

Using the Stage 3.2.2 simulator (Vaughan, 2008) and DTU MobotWare (Beck et al., 2010) as controller, a quantitative evaluation was conducted to investigate the characteristics of the narrow passage detection model. The aim of the experiment is to investigate how tolerant the detector was against variations in approach angle and offset within the narrow passage. Furthermore, I want to investigate the possibilities of tracking and prediction of the situation.

A simple simulation environment was set up with two rooms, separated by a long wall with a 60 cm wide doorway connecting the rooms. The robot used for the simulation was a simulated version of the DTU SMR robot, which is 30 cm wide. The relatively narrow doorway of only 60 cm was chosen to represent a door width which poses a challenge for the small robot. Figure 7.4 shows images from the simulation environment when the robot performs a doorway passage.

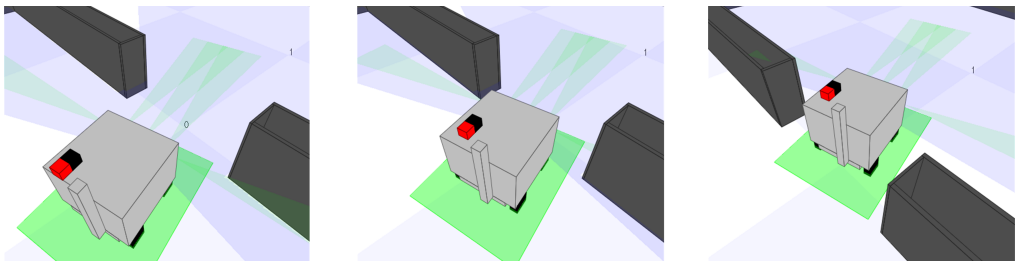

Figure 7.4: Simulation environment for examining situation modeling and recognition of passing through a narrow doorway. The simulated DTU SMR robot is 30 cm wide and moves between two rooms through a 60 cm wide doorway.

A simulated test sequence was conducted where the robot passed through the doorway 20 times, starting at location offsets and approach angles which were randomly selected in the range $x \in [-0.7; -1.0]$, being the distance from the doorway, $y \in [-0.25; 0.25]$, being the offset from the center of the doorway and $\theta \in [-40°; 40°]$ being the approach angle. If a set was chosen which would collide with the doorway, a new random set was drawn. Figure 7.5 shows the robot trajectories throughout the simulated sequence.

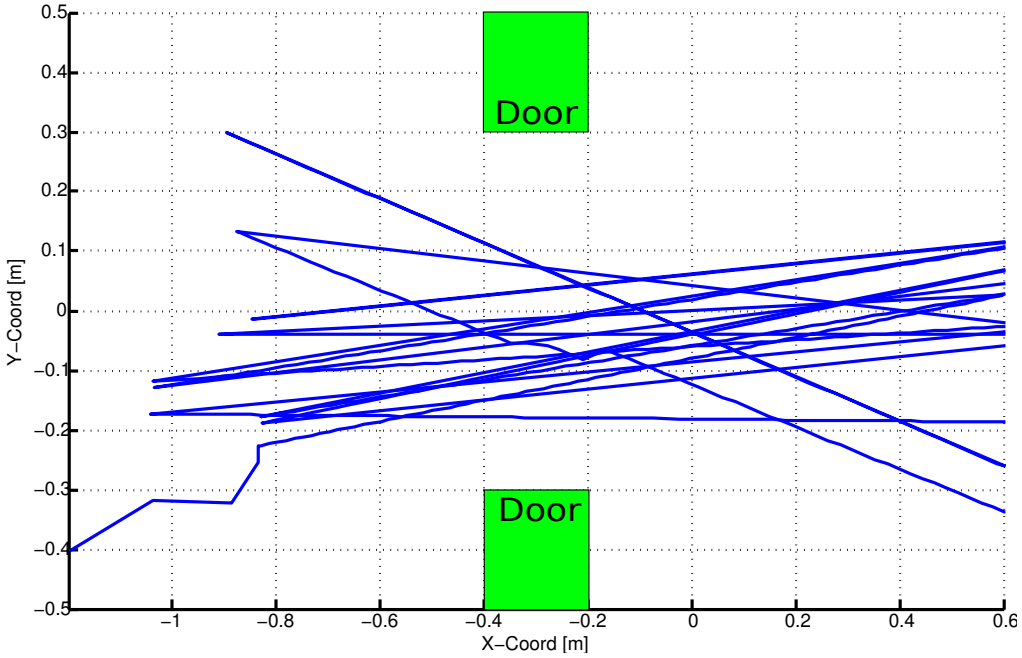

Figure 7.5: Trajectories of the robot during the simulated test sequence.

The simulated experiment was initialized without any predefined states in the EMM, neither from the situation model or binary loaded. The first pass through the passage generated the initial state sequence in the EMM model. This model was then subsequently used for matching the states of the remaining simulated runs. The Jaccard similarity measure was used for cluster matching, using a similarity threshold of $\lambda = 0.99$.

New states in the EMM model are identified by consecutive IDs, starting at 1. When the robot made the first pass through the door, a state sequence from 1-11 was generated before the robot was inside the doorway. When the doorway has been passed by the front of the robot, the laserscanner no longer sees the doorway, which causes various other features to be detected and a number of new states created. Figure 7.6 illustrates the trajectory of the first passage through the doorway, including the points where the EMM transitions to a new state.

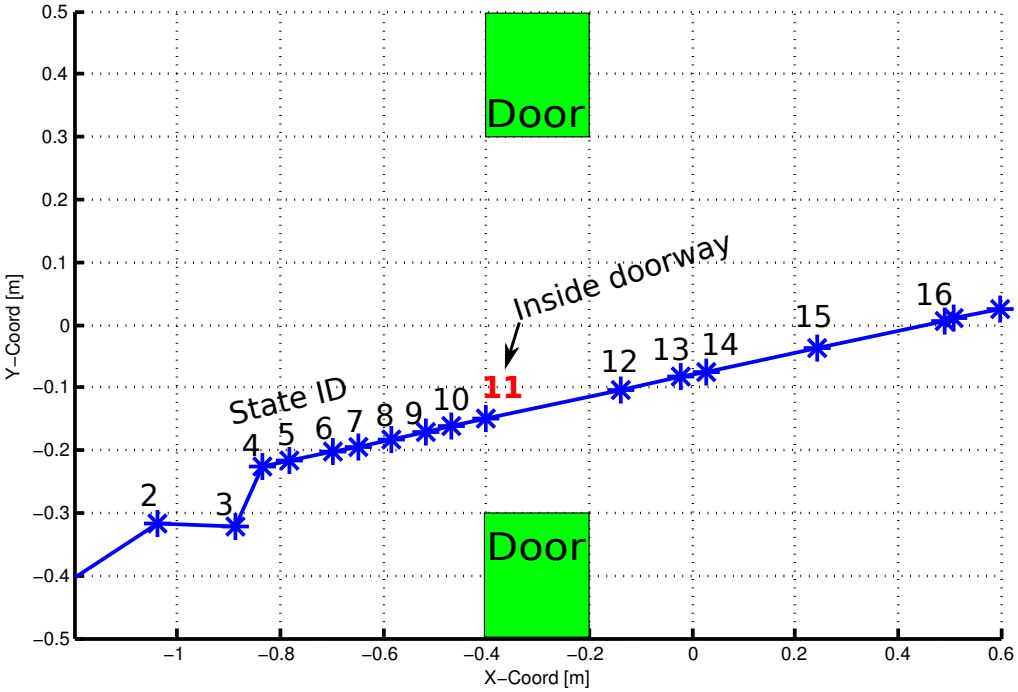

Figure 7.6: Trajectory of the first passage of the robot. The points where the EMM transitioned to a new state are annotated by stars and ID. State 11 accounts for the point inside the doorway.

Following the first pass through the doorway, the last 19 passes through the doorway were run using the same EMM model to evaluate the capabilities of matching states to recognize and predict the situation. Figure 7.7 shows the total set of robot trajectories together with the contours of EMM states that were matched through the passes. For plotting purposes, the state contours have been interpolated between the discrete states which follow the trajectories, which cause some noisy edge effects around the top and bottom of the plot.

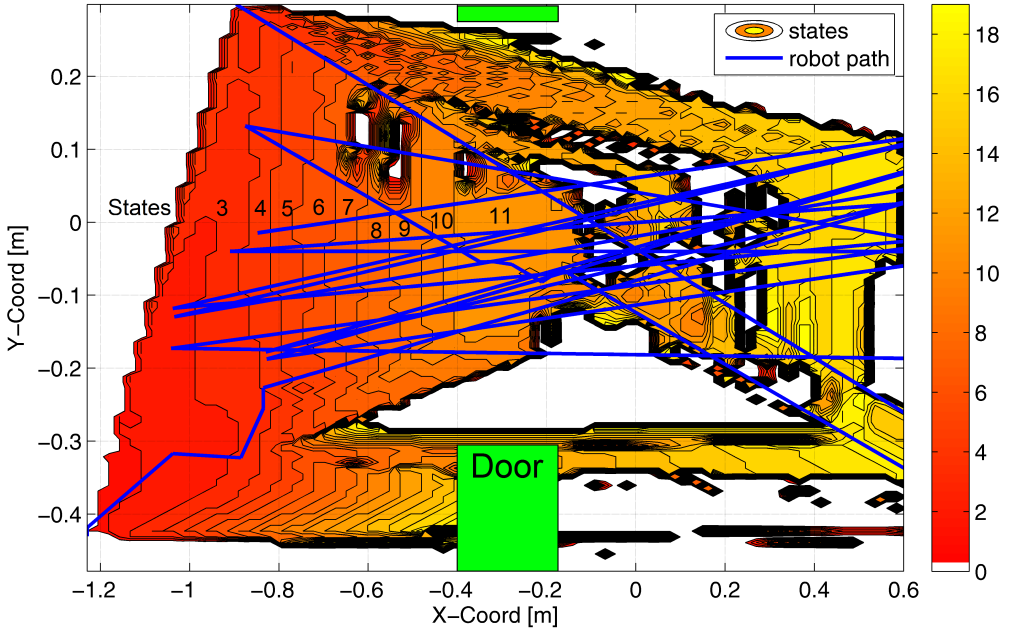

Figure 7.7: Visualization of the robot trajectories and situation state regions. Region noise is primarily caused by plotting interpolation between measurement points rather than state-matching noise. White areas are regions that have not been classified within the 18 first samples and have been removed for plotting purposes.

Each color layer in figure 7.7 represents a region where the robot is estimated to be the same state. Clearing the plot from robot trajectories, figure 7.8 clarifies the state transition diagrams. Observe how the situation assessment follows a similar sequence of states for each pass of the robot, except for a few miss-matches around the (-0.6,0.2) region. When the robot has passed through the door, the state sequence gets less predictable, as the laserscanner now sees the closest obstacles on the other side of the passage.

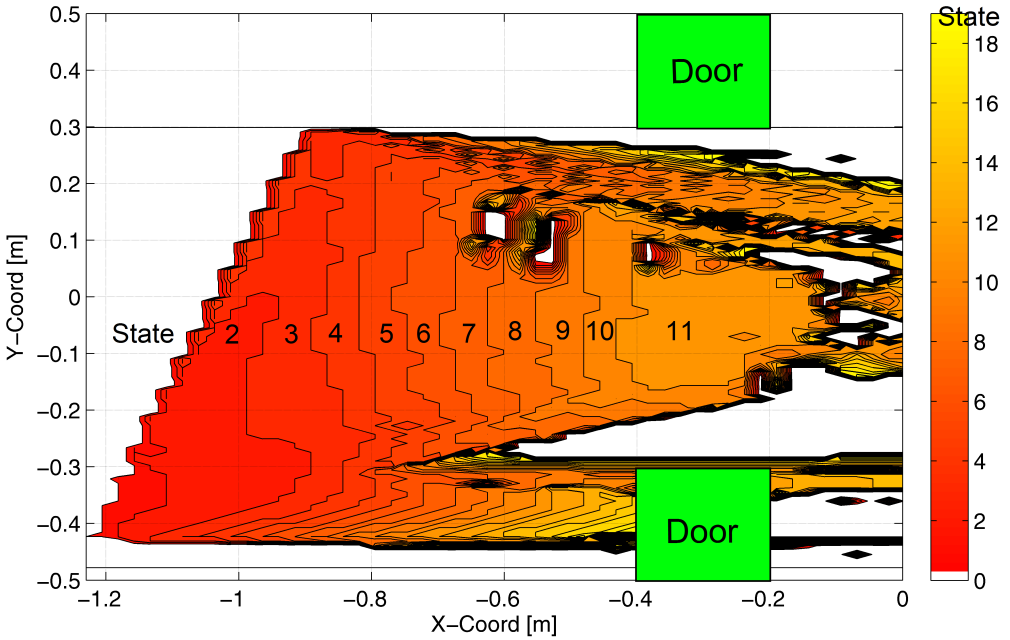

Figure 7.8: Close-up on state regions without the robot trajectories. It is clearly evident that each robot pass has followed a similar sequence, except for a few spurious states around (-0.6,0.2) that have caused steep gradients in the interpolation.

Already from a significant distance, the state sequence predicts a high likelihood of a narrow passage. For example, observe in figure 7.8 how the state sequence is almost linear from 60 cm before the door and until it has been passed. State 11 is observed to be the center of the door, with a state data-set of

$$s_{11} = \begin{cases} d = 0.61 \text{ m} \\ a = 3.09 \text{ rad} \end{cases} \tag{7.7}$$

Where the combined distance from the laserscanner equals the width of the door, and the angle difference between the closest points is approx. 180 °. Following the state transitions up to state 11 in the state transition likelihood matrix, reveals a clearly identifiable sequence back to state 4. To make a robust model, a slightly smaller situation sequence is chosen to represent the narrow passage by the following state sequence X_s:

$$X_s = \{7, 8, 9, 10, 11\} \tag{7.8}$$

The transition maximum likelihoods for the states of X_s, as calculated by the prediction engine, are shown in table 7.1.

State	Transition to	Likelihood
7	8	1.00
8	9	0.95
	7	0.05
9	10	1.00
10	11	0.91
	12	0.09
11	12	0.55
	14	0.05
	34	0.30

Table 7.1: State transition likelihoods for the states of the level 2 situation model X_s throughout the simulation.

The state transition table 7.1 can also be visualized in a directed graph, where the likelihood of predicting a passage through the narrow space is clearer.

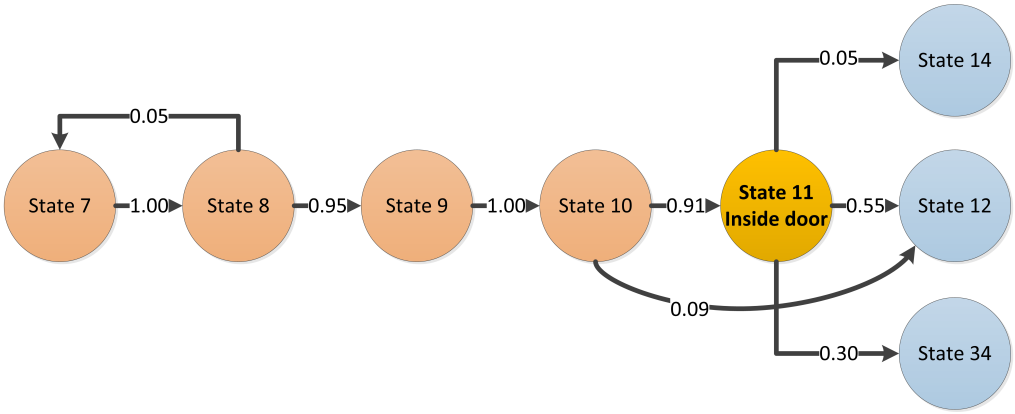

Figure 7.9: Summary of the state transition likelihood for passing through the door. Through the 20 simulated runs, prediction through the chain is very high.

The typical sequence has been identified by inspection as of figure 7.9, so the prediction accuracy can be calculated from the Markov chain by its maximum likelihood, given by

$$L(s_{11}|s_7) = \prod_{i \in X_s} a_{i,i-1} = 0.86 \tag{7.9}$$

This can be considered as a firm ground for changing to a different motion planner, which is optimized for careful navigation through the passage. In opposition to similar work for example by Meyer-Delius et al. (2009a), both situations

and state space are learned from actual operation in the environment. By the flexibility of the approach, situations and states can be classified during on-line operation, pre-imposed by hand-crafted models or trained from a data-set.

In this experiment, I have shown that a level 1 perception model can be defined without too much complexity to produce meaningful situation states. By classification, a level 2 comprehension model X_s is extracted, that robustly detects the planned situation. FinallyÂ¨, it was shown that a level 3 prediction model can be derived to allow the use of transition likelihoods for situation prediction.

7.1.3 Door passage using real world data

Using a small mobile SMR robot platform, the framework was also evaluated in a real world setup. The robot is similar to the simulated robot in figure 7.4 and is also running the DTU MobotWare mobile robot control software. For sensing the narrow passage through the door, the robot was equipped with a Hokuyo URG-04LX laserscanner mounted 5 cm above the ground and configured to cover 180°in front of the robot. The scanner samples the distance for every 0.35°at a frequency of 10 Hz. Using this set-up, the robot was navigated through two laboratory doors and one office door.

Figure 7.10: Small module robot passing through a doorway at a common office in DTU Building 326. A passage through 3 different doorways was investigated in this experiment.

Like in the simulated tests, the first passage through a laboratory door generated the states that subsequently were used for matching. A Jaccard similarity threshold of $\lambda = 0.99$ was used for cluster matching to keep results comparable to the simulated experiment. Unfortunately, it is not possible to use the geographic position of the robot as a comparison measure in this test, as the environments were different and only odometry was available. Figure 7.11 shows the state transition time line for passing through the doors.

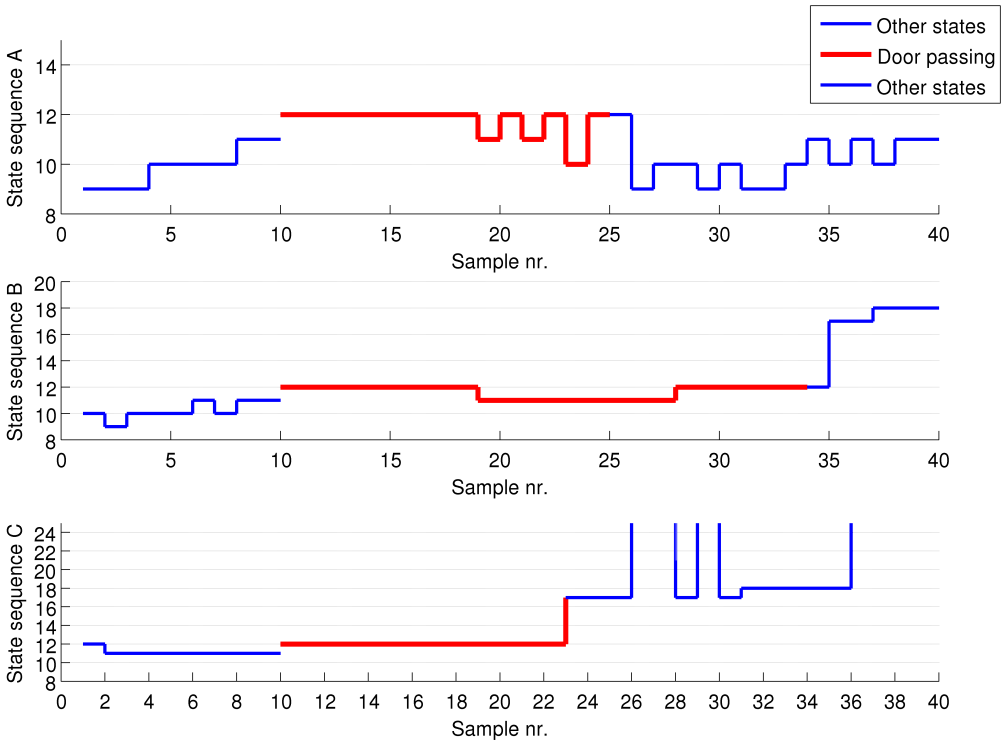

Figure 7.11: State transition time-line for passing through two lab doors (A and B) and an office door (C) in a traditionally cluttered university environment.

Samples 10 to 18 in figure 7.11 (in red) are the actual passage through the doorway, that is clearly identified in all three situations as state 12. Some state noise is present, especially when the robot has passed through the doorway, as we also saw on the simulated tests.

Very noteworthy is that state 12 is preceded by state 11 in all three cases. Examining the state transition matrix concludes that the likelihood for transition from state 11 to state 12 is

$$P(s_{12}|s_{11}) = 0.56 \tag{7.10}$$

Although the likelihood is not completely conclusive, it is still useful for predicting possible narrow passages ahead and i.e. for selecting a slow maneuver speed or a motion planner with focus on keeping the center between obstacles. The last part of the red sections is passage past the open door and objects cluttering the doorway, creating noise but still some detections of state 11 and 12 indicating a narrow passage ahead or at the current locating.

In this experiment I have shown that a robust detection of being within a narrow passage is also possible using real data and parameters identical to the simulated experiment. Prediction is not possible through as many states as in

the simulated experiment, but performance might be improved by optimizing the similarity threshold or selecting a more advanced clustering strategy that handles sensor and environment noise better.

7.2 The AGV Logistics Robot

At DTI we have devoted quite some attention to the trends of AGV logistics robots and how they are becoming more and more common in modern industry. From being an isolated transportation system in closed warehouses, innovation efforts have now matured AGV application for operation in open areas among factory and warehouse workers. To demonstrate this, an AGV demonstration was set-up at the DTI robot laboratory to transport material carts between robot installations as illustrated in figure 7.12.

Figure 7.12: Demonstration platform for the AGV demonstrations at the DTI Robot laboratory.

The mobile robot platform used for the coffeebot experiments was retrofitted with an electromagnetic coupling (powerful door magnet) to connect the transfer carts, which proved to be a mechanically simple way of attaching loads in excess of 500 kg. Figure 7.13 illustrates the magnet device on the AGV platform. Detailed positioning in relation to the carts was done by placing a checker-board based guidemark at the front of the cart (as seen in figure 7.12). The guidemark module was designed for MobotWare by Andersen (2005) and makes it possible to estimate the relative position between camera and guidemark in 6 degrees of freedom. Position estimation accuracy was achieved down to an average of 0.2 cm by Andersen (2005) with well calibrated camera geometry. Using the Microsoft Kinect camera and an automatic tilt-calibration module by Viager (2012) an accuracy of below 2 cm is achieved only by measuring the camera offset from robot base in (x, y, z) coordinates. The electromagnetic coupling is quite sensitive to full surface contact before it develops proper holding force but the accuracy of the guidemark positioning approach using the automatically calibrated Kinect camera has proven adequate to robustly pick up the cart. Besides being useful for positioning, the guidemark also contains a 10-bit ID-

code, which is used for identification of the carts.

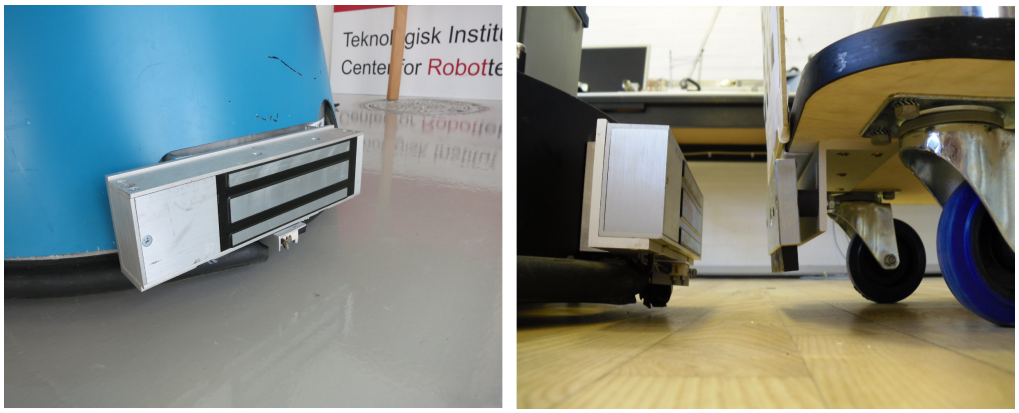

Figure 7.13: The AGV platforms at DTI and DTU were retrofitted with a magnet-based coupling to electronically attach and release carts.

In order to investigate the versatility and transferability, a parallel demonstration was setup at the laboratory of DTU Electrical Engineering, Automation and Control by Viager (2012). The purpose of the two parallel demonstrations is to investigate the development of skill-based modules for an AGV and how they can be transferred between several physical setups. The platform for the DTU setup is a twin of the robot used for the DTI setup, using similar carts and electromagnetic coupling.

The aim of this demonstration in context of this thesis is to investigate the benefits of using situation assessment for a mobile robot application to detect and possibly predict critical situations and errors. Most traditional research applications for mobile robots deal with exploring and moving along unique routes which are constantly re-planned. This demonstration targets the repetitive operations which are common for commercial AGVs by just moving carts between two locations.

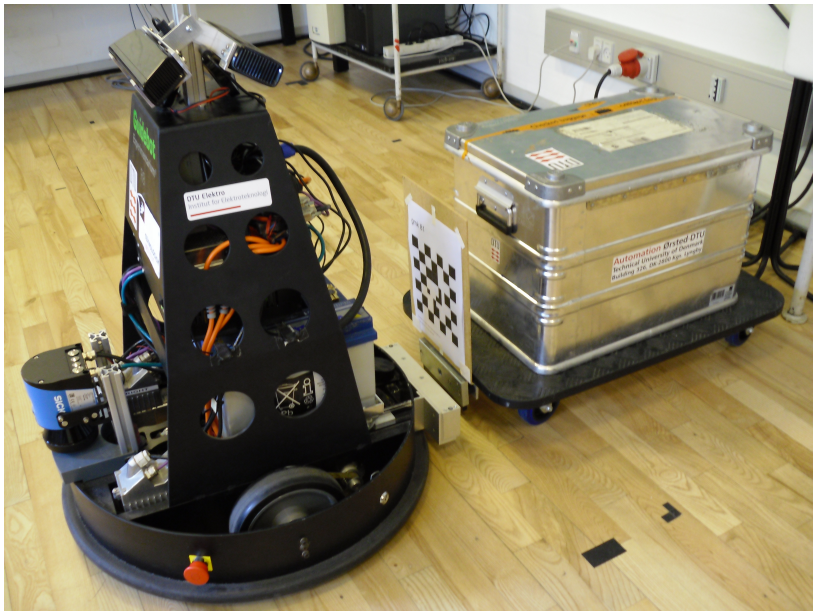

Figure 7.14: The twin AGV demonstration platform at DTU Electrical Engineering, Automation and Control laboratories. The data-sets used for this experiment were collected from this platform.

7.2.1 Situations in the application

The complete scenario for this demonstration follows a sequence, where the AGV is initialized at a start position. From there, it moves to cart location 1 and picks the cart. The cart is moved and dropped at location 2, whereafter the AGV moves to a pause location. After a short pause, the AGV moves to pick the cart at location 2 and returns it to location 1. Finally, the AGV moves to the start position, where the mission loops.

Mission control is implemented through a state-machine in an AURS rule script. The script coordinates script-based modules for initialization of the robot localizer, movement between start, cart 1, cart 2 and pause locations and invocation of rule-based cart pick-up script. The state-machine follows the sequence of states m_{state} as shown in table 7.2

State m_{state}	Action	Cart attached
1	Initialize EKF localizer from guidemark	No
2	Move from start to cart 1 position	No
3	Execute cart pick-up rule	No→Yes
4	Move to cart 2 position	Yes
5	Execute cart drop-off rule	Yes→No
6	Move to pause location	No
7	Re-initialize localizer from guidemark	No
8	Move to cart 2 location	No
9	Execute cart pick-up rule	No→Yes
10	Move to cart 1 position	Yes
11	Execute cart drop-off rule	Yes→No
12	Move to start location	No

Table 7.2: State sequence of the AGV Application demonstration.

Initialization of the MobotWare EKF localizer is done by using the guidemark pose estimation module, which also is used for docking to the carts. 2 guidemarks are placed at fixed locations near the AGV start and pause position, and their absolute position is integrated in the AURS Mapbase module. By estimating the robot location in relation to the guidemark coordinates in state 1 and 7, the robot is capable of self-initializing the localizer as long as it is within visible range of the guidemark.

Figure 7.15 illustrates a plot of robot trajectories in localizer map-coordinates when performing a set of 13 loops through the mission sequence. The plot also illustrates the start, pause and two cart locations. Furthermore, the plot has been annotated with the mission states and movement directions to clarify which parts of the sequence belong to what mission state.

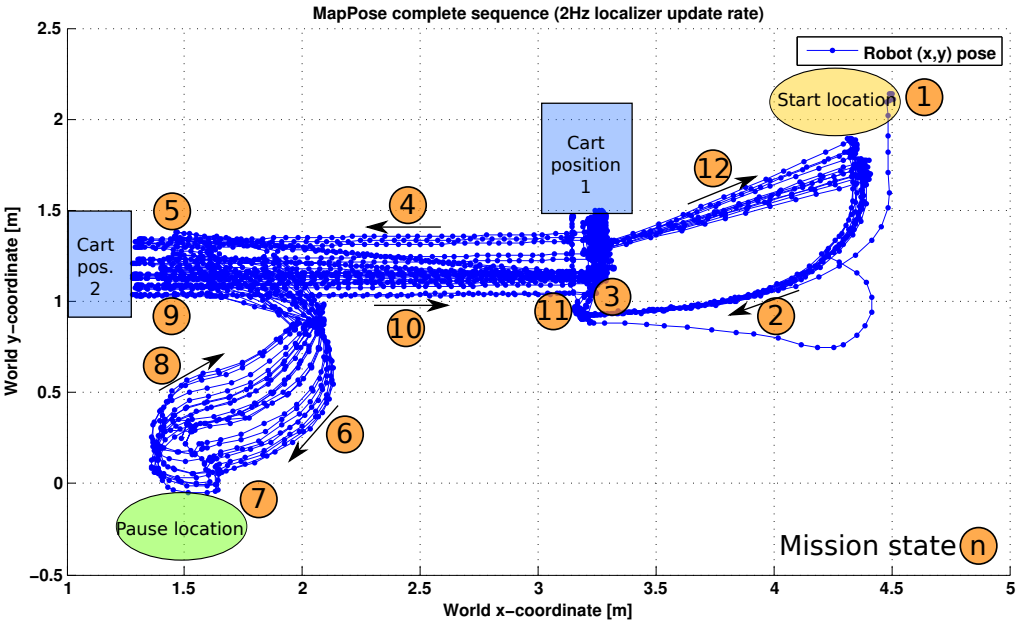

Figure 7.15: Robot trajectory plotted through the large 13-loop demonstration scenario. The robot begins at the start position, picks the cart at location 1, moves it to location 2 and then moves to the pause location. After a short while, the robot returns to pick the cart at location 2 and moves it to location 1 and then it returns to the start location.

What is interesting to notice in figure 7.15 is that there is only little overlap of the trajectories between the loops. Especially the motion in state 4,10 and in state 6,8 has a large variation in the trajectories. Even the pick-up and drop-off of the cart in location 2 varies by almost 0.5 m at location 2, which works without problems, as the pick-up is guided by local estimation of the cart pose using the guidemark on the cart.

The large variation in robot motion trajectory makes it challenging to construct reliable spatio-temporal models of the robot motion itself, as the similarity threshold must be set low enough to suppress the variations in trajectory, which makes it difficult to capture any deviations from sequence before it might be too late.

In the following experiments in this section, I investigate design of situation models designed to capture the characteristics of the AGV logistics robot application to capture errors early and to aid control system with prediction of the future behavior of the robot.

7.3 Detection of errors in AGV localization tracking

The primary source to global localization in DTU MobotWare is an EKF based localizer, which uses a RANSAC process to extract line-based features from a laserscan of environment contours. These lines are matched against a static map of expected lines in the environment to continuously track the global location of the robot. This approach to localization is fairly robust, as the maps are manually measured and can be tuned for best possible visibility for the robot. Nevertheless, being an incremental algorithm, Kalman filtering-based localizers are vulnerable to wrongful matches or too many measurements without any match, as the filter often is incapable of recovering the correct position when position tracking is lost. In MobotWare the most common source of application errors is the localizer loosing track of the robot position and thus crashing the robot.

In this example I investigate a section of the AGV logistics application at DTU, where the robot performs 3 pick-up operations of the cart. In the end of the third iteration, the robot looses localizer tracking of the position and stops. The situation type addressed in this experiment is the Type 2: deviation from known situation pattern, where the initial correct behavior of the robot is used to form the baseline of normal operation and significant deviation from this is treated as emergence of a critical situation.

Figure 7.16 illustrates the robot trajectory through the demonstration as calculated by the localizer, following the scenario described in section 7.2.1 on page 125. Notice that the robot is following different routes to and from the start and pause locations due to the different motion commands used, which will be clearly visible in the later spatio-temporal analysis.

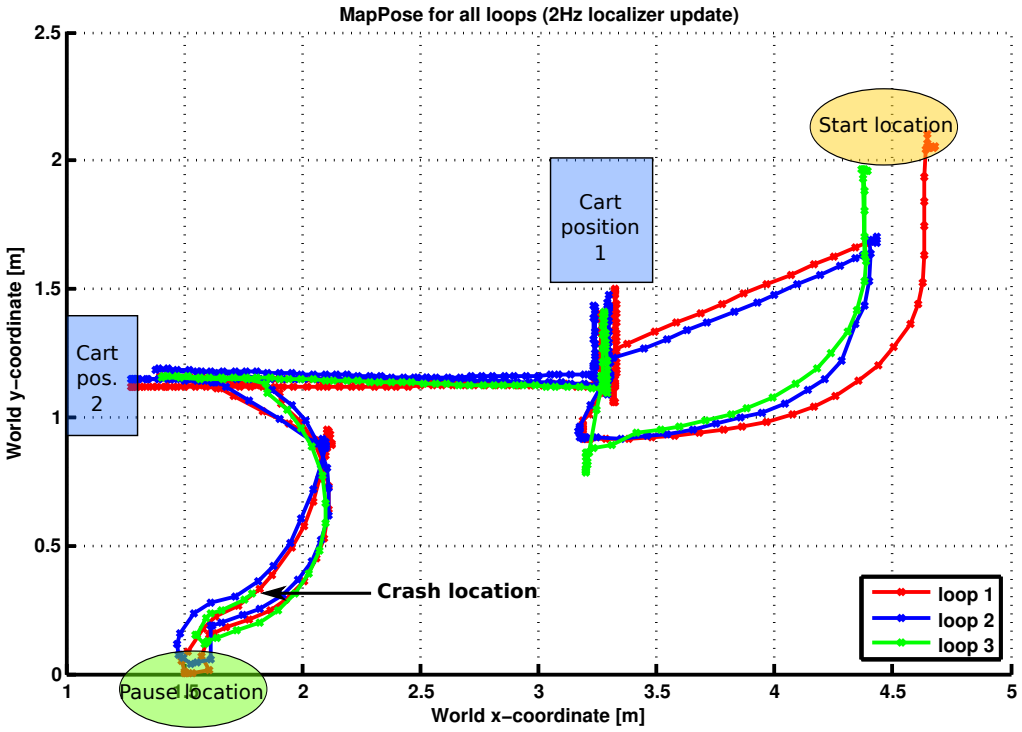

Figure 7.16: Robot trajectory in the evaluation scenario. During the third loop, the robot looses localization and stops after the pause location.

Observing the robot trajectory itself does not reveal that anything bad happens at the third loop, as the robot does not move significantly out of its normal path before it is stopped in this experiment. Other means are required to identify the localization error which can be found in the EKF covariance matrix Σ_t.

The covariance matrix Σ_t represents the robot location uncertainty at a given time t. When the robot moves, position uncertainty grows due to the accumulating measurement noise of the robot odometry. Through the EKF correction the noisy position estimate is matched to the known line-features of the map and thus position uncertainty is reduced. High values of the covariance matrix also increase the probability for wrongful association as associations depend on the certainty of robot position. These characteristics of the EKF covariance matrix make it obvious to include it in a situation model which monitors localization performance.

Figure 7.17 illustrates the first diagonal element (1,1) of the 3 x 3 covariance matrix Σ_t which represents the uncertainty with respect to the x coordinate of the robot. Covariance for each of the three loops follows similar patterns, where covariance grows at certain positions, where no localization matches provide correction information in the x direction. The other diagonal elements represent similar uncertainties in y and θ. When the localizer looses track of the robot position in the third loop, it becomes visible by a jump in the covariance immediately. Other peaks in covariance are caused by re-initialization of the localizer, where an initial value of 0.05 is used.

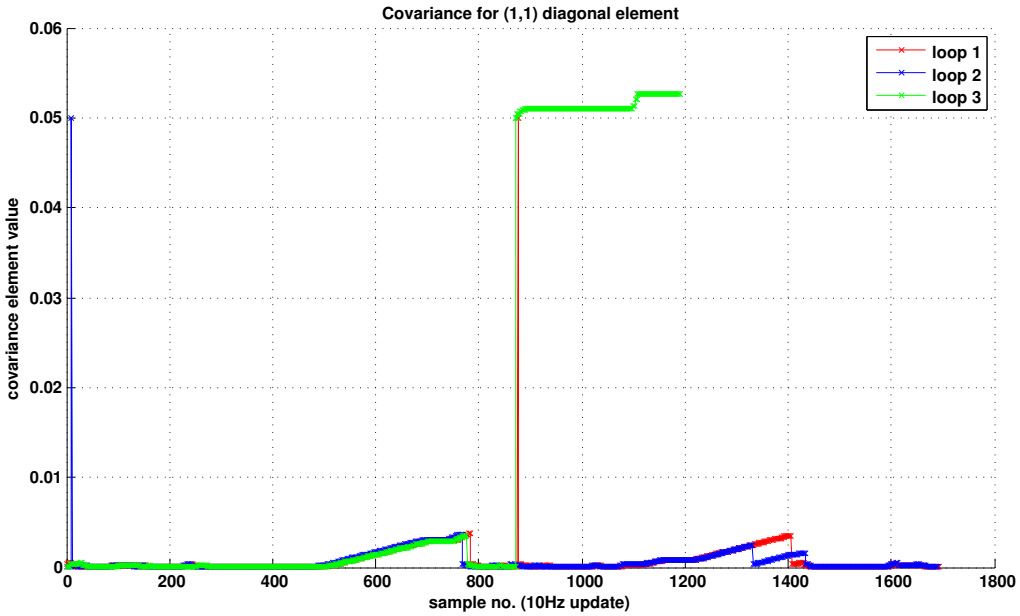

Figure 7.17: Covariance element $\Sigma_t(1,1)$, representing the uncertainty in relation to the estimate of the robots x coordinate. In certain areas no measurements provide information for the x dimension, where the covariance steadily grows. When the localizer fails in third loop, it is clearly visible by a jump in the covariance.

7.3.1 Situation model design

Situation assessment is essentially the matter of evaluation of relations between entities, their characteristics and behavior within a given context. In order to design a situation model which can capture the characteristics and behavior of the AGV application, the (x, y) coordinate of the map pose from the localization system is a natural choice. As I am interested in capturing and possibly predicting eventual errors in the localization system, the relations between the localization pose (x, y) and localizer covariance Σ_t will add the information about localizer estimation quality, to reveal lost pose tracking.

A situation model was designed, containing the following elements:

$$s_n = \begin{cases} x_{map}, y_{map} & \text{x,y coordinate from map pose.} \\ diag\,(\Sigma_t) & \text{Diagonal elements of 3x3 covariance matrix} \end{cases} \tag{7.11}$$

No initial pre-defined states are included in the model as the start and pause locations as well as the route have no clear definition of exact location and varies throughout the number of loops. Cart locations are unknown to a larger extend than being within a pick-up zone, where the fine-positioning is performed by the Kinect camera and the guidemark.

7.3.2 Situation assessment results

Running the experiment through SA-Platform, using the Jaccard distance metric with a similarity threshold of $\lambda = 0.995$, an EMM spatio-temporal model is constructed. As a key variable component of the situation model is the geographic (x, y) coordinate of the structure, the EMM structure follows similar patterns as the robot trajectory.

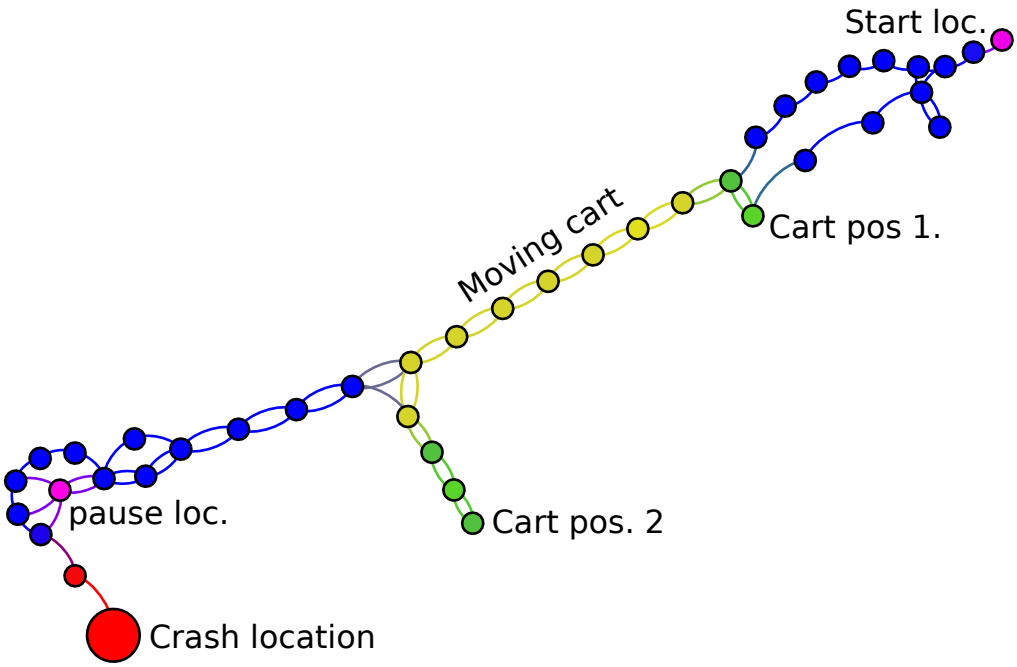

Figure 7.18: Constructed EMM spatio-temporal model of the AGV application. Notice that nodes are not positioned geographically but as a result of an on-line force-based layout algorithm, although it has been repositioned to have somewhat similar structure to the robot trajectory in figure 7.16.

The resulting Markov Chain is illustrated in figure 7.18, where the repetitive cycle of movement between start position, cart position 1 and 2 and the pause position has formed the structure of the chain. Loops in each end of the chain are caused by the robot moving in different routes on the way to and from the start and pause positions.

The critical event of the localizer loosing position tracking at the end of the third loop is identified and visualized by the red nodes in the left part of the graph. The event is clearly identified as a Type 2 situation as a deviation from the known sequence with an abnormal relationship between robot position (x, y) and covariance Σ_t.

Through on-line visualization in Gephi an operator can easily follow the current state of the robot through the state sequence of the EMM and have improved situation comprehension. The large red node in figure 7.18 marks the current node (which is the state of crash in this example) and makes it easy for the operator to understand where in the sequence the robot is, and what the

next moves are expected to be.

Adding the transition likelihoods to the edge weighting in figure 7.19 makes it visible which parts of the sequence are parts of the core sequence for the robot, and which are considered rare states, created by deviations from the core patterns. The deviations are caused by different motion patterns around the start and especially pause locations.

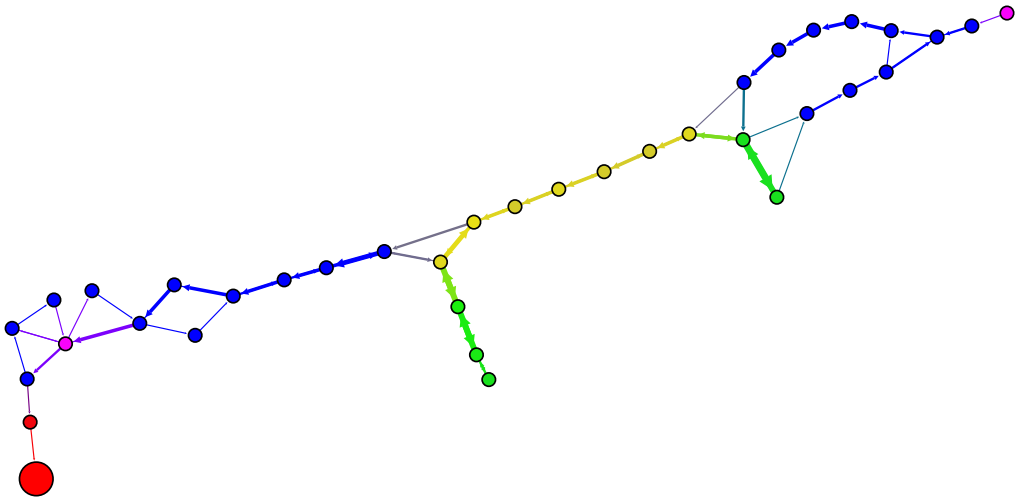

Figure 7.19: The EMM model of the AGV application sequence with edges weighted by the likelihood of them being traversed.

The weight of edges in figure 7.19 corresponds to the number of times the edges have been traversed during the on-line construction of the model. Already after the three loops through the sequence, it is fairly possible for an operator to predict the future states from the on-line visualization. Before it makes sense to look into automated prediction using the prediction engine, it is necessary to look into a larger data-set.

7.4 AGV Situation characterization and prediction

AGV logistics robot installations are not limited to the normal 8-hour shifts of their human counterparts. But in order to have gain from this benefit, the robot installations must be reliable in long term operation and capable of handling the critical situations.

In this experiment I investigate a larger data-set of 25 minutes operation of the AGV application performing 13 loops through the cart moving sequence, as illustrated earlier in figure 7.15 on page 127. The data-set is applied to construct a baseline EMM model for analysis of Type 1 situations (detection and prediction of known future events).

The purpose of this experiment is to investigate the performance of the prosed framework for situation assessment on a large data-set with repetitive mobile robot behavior, as it is a common scenario for commercial AGV systems. The

initial challenge is to design a comprehensive situation model, which captures the details and structure of the AGV application. The situation model will be applied in the SA-Platform to construct a spatio-temporal EMM model of this large data-set. With basis in the reference model for human situation awareness by Endsley (1995) I analyze and discuss the situation assessment results according to the three level of SA: perception of elements in the situation, comprehension of the current situation and prediction of future status.

This sequence eventually also ends in a localization error in the beginning of the 14th loop, but this detection is not the prime target of this investigation. Nevertheless, the event is actually captured by the model and is used to evaluate the performance in handling Type 2 situations (deviation from known patterns) as well as methods of capturing random "black swan" events.

7.4.1 Situation model design

In order to obtain the first level of situation awareness, it is necessary to select the elements in the environment which defines perception of the situation. The elements should capture the essential characteristics of the situation, so that states of the EMM model have a descriptive meaning in the situation, and yet not so much data, that the situation picture becomes occluded.

The situation model to capture the motion behavior of the robot does again include the map pose from the localizer, but this time the entire pose of (x, y, θ) has been included to enable the spatio-temporal model to also capture any rotational motion.

When moving the cart between the two locations an important characteristic is the input, i_{switch} from the micro-switch, which detects if the cart is correctly attached to the electromagnetic coupling. Including this value in the situation model will make state sequences depend on if the cart is attached and clearly identify errors if cart attachment fails or the cart are lost during movement.

The robot behavior depends on the mission state m_{state}, so including the mission state in the situation model makes the model reactive to any robot behavior which does not fit into the correct state as well as motion through the same map poses in different states becomes separated.

The diagonal elements of the localizer covariance matrix Σ_t include the localizer prediction quality into the situation states, which are crucial to detect localization errors early as seen in the earlier example in section 7.3 on page 128 and detect significant changes, e.g. in the environment, which affect localization performance.

Combining these 8 attributes result in a situation model as defined in equation 7.12:

$$s_n = \begin{cases} x_{map}, y_{map}, \theta_{map} & \text{(x,y,}\theta\text{) Coordinate from ma ppose} \\ diag\,(\Sigma_t) & \text{Diagonal elements of 3x3 covariance matrix} \\ i_{switch} & \text{Micro-switch state (0/1)} \\ m_{state} & \text{Mission state} \end{cases} \tag{7.12}$$

This situation model clearly demonstrates the strength of the heterogeneous DataSource module in SA-Platform. A XML implementation of the s_n situation model, as shown in listing 7.1, quickly connects the required data-source components to the SA-Platform which combines the input values into one streaming data-set for processing by the clustering and EMM engine. A Jaccard similarity threshold of $\lambda = 0.97$ is selected to determine if new data-sets should be assigned to existing clusters or new should be created.

```
<emm name="AGVApplication" >
 <model name="AGVMotion">
  <clustering method="Jaccard" threshold="0.97"/>
  <probability enable="true"/>
  <data source="aurs" variable="mappose.pose" sync="true" index="0
      1 2"/>
  <data source="aurs" variable="localize.covar" sync="false" index=
      "0 4 8"/>
  <data source="rhd" variable="digital" sync="false" index="0"/>
  <data source="aurs" variable="global.dockdemo.state" sync="false"
       index="0"/>
 </model>
</emm>
```

Listing 7.1: XML situation model to configure the DataSource in the SA-Platform to combine localizer variables and mission execution states from AURS together with digital input from the Siemens PLC on the Guidebot platform via RHD.

This situation model has been designed without performing any pre-processing on any of the input variables. If a higher level of abstraction was desired in the situation model, variables of high variance or improper reference, such as $x_{map}, y_{map}, \theta_{map}$ or Σ_t, could be classified into categories (e.g. high/low covariance) or transformed into more suitable values by pre-processing scripts. The clustering engine were specifically designed to allow use of raw and noisy values, as they are represented in real-world robot systems, which have been the motivation to design this situation model only using raw values directly from the robot control framework.

Similar to the narrow passage detection example in section 7.1 on page 110, the first loop through the sequence creates the initial EMM sequence of states and the consecutive loops are matched to these states or form new states if similarity drops below the Jaccard threshold. If a certain data-points was well known and had a certain interpretation in the application, they could have explicitly been introduced in the situation model as an initial point. In this scenario no exact procedure is defined for the AGV and thus no initial data-points are introduced through the situation model.

7.4.2 Situation assessment results

According to the situation awareness model by Endsley (1995), the formation of situation awareness is a process in three levels. In the following section I investigate the results of using the designed situation model on the large data-set to obtain these three levels of situation awareness in relation to the AGV logistics application.

Level 1: Perception of elements of the situation

Perception of the situational elements in the AGV application is investigated by analysis of the resulting graph structure after running the situation assessment process in the SA-Platform. Key performance criteria are to evaluate if the graph describes the application in adequate detail to identify the process but also that the relationships between situation variables does not create large and noisy graphs without any clear structure.

The EMM model is constructed by processing the 25 minute long data-set and a total of 20.499 data-points were clustered and tracked through the spatio-temporal EMM model. The data-reduction capability of stream-based clustering was proven by a reduction to an EMM model with only 48 clusters (nodes), which were connected by 58 directed edges as shown in table 7.3.

EMM data-set	
Nodes	45
Edges	58

Table 7.3: The large AGV data-set resulted in an EMM model with 48 nodes, connected by 58 directed edges. It was a significant reduction from 20.499 sets of raw data.

Visualizing the graph, where nodes are positioned by their geographic (x_{map}, y_{map}) pose as in figure 7.20, makes it easier to examine the EMM states in relation to the raw robot trajectory plot in figure 7.15 on page 127. The motion pattern of the robot is clearly identifiable although the density of states varies.

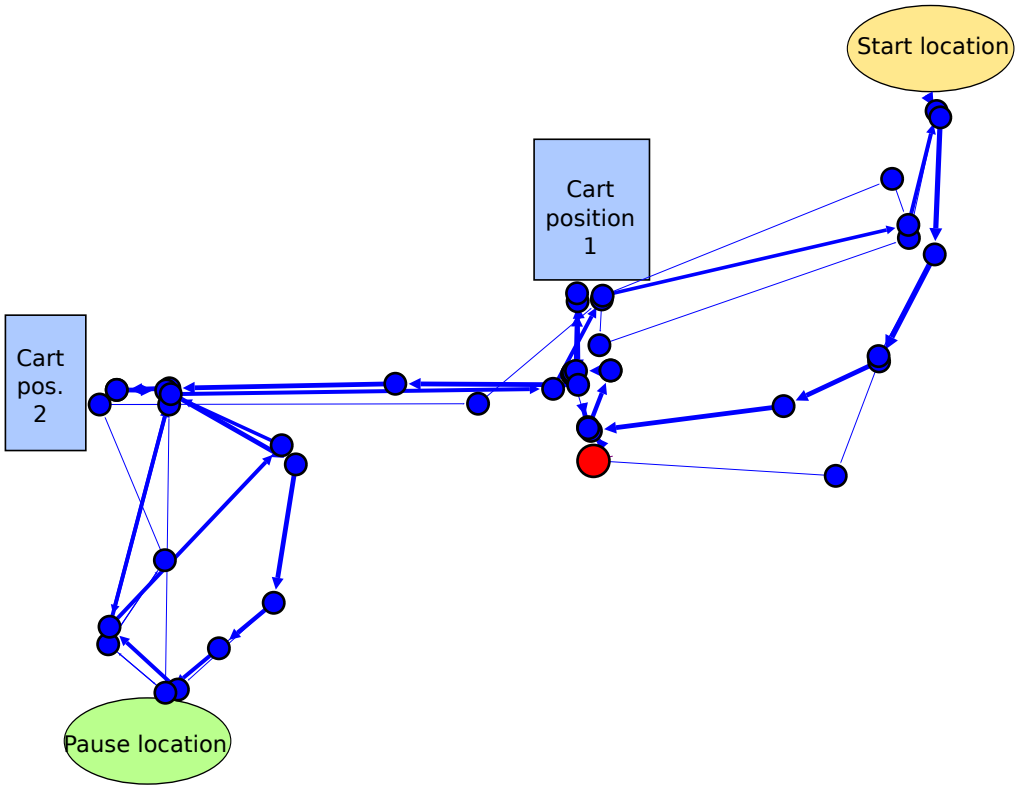

Figure 7.20: Geographic representation of states in the EMM model for the large AGV dataset. Most loops through the experiment follow the same sequence in the graph. Width of edges represents how many frequently they have been traversed. As the robot did not follow the same trajectory through all loops, alternative nodes to the main sequence are created especially in the left side, as there was large variation in robot trajectories. Many states are overlapping geographically, as they only differ in θ_{map} orientation or m_{state}.

Figure 7.20 shows, that the (x_{map}, y_{map}) pose only represents a small part of the situation model in opposition to the earlier experiment in section 7.3 on page 128. Regions with only linear robot motion between cart position are sparse in the graph, as only (x_{map}, y_{map}) changes significantly in this region. Regions around pick-up, drop-offs and pause location is dense, as there are much θ_{map} rotation as well as changes in digital input i_{switch} and m_{state}. The geographic layout of the graph makes the graph-structure comparable to the robot trajectory plots but also occludes the true structure of the graph. Figure 7.21 depicts the same EMM graph after running a force-based layout algorithm, color-coding the nodes according to m_{state} and adding node ID numbers.

The force-based layout emphasizes the structure in the graph which is connected by edges of high likelihood. In figure 7.21 it reveals that the central part of states in the EMM model follows a circular pattern. Some branches diverge from this pattern, which is caused by the variation of robot trajectories. Diverging branches in the model only improve the performance in recognition of variations in the situation, as long as branching nodes do not appear spurious and without creating any real new structure.

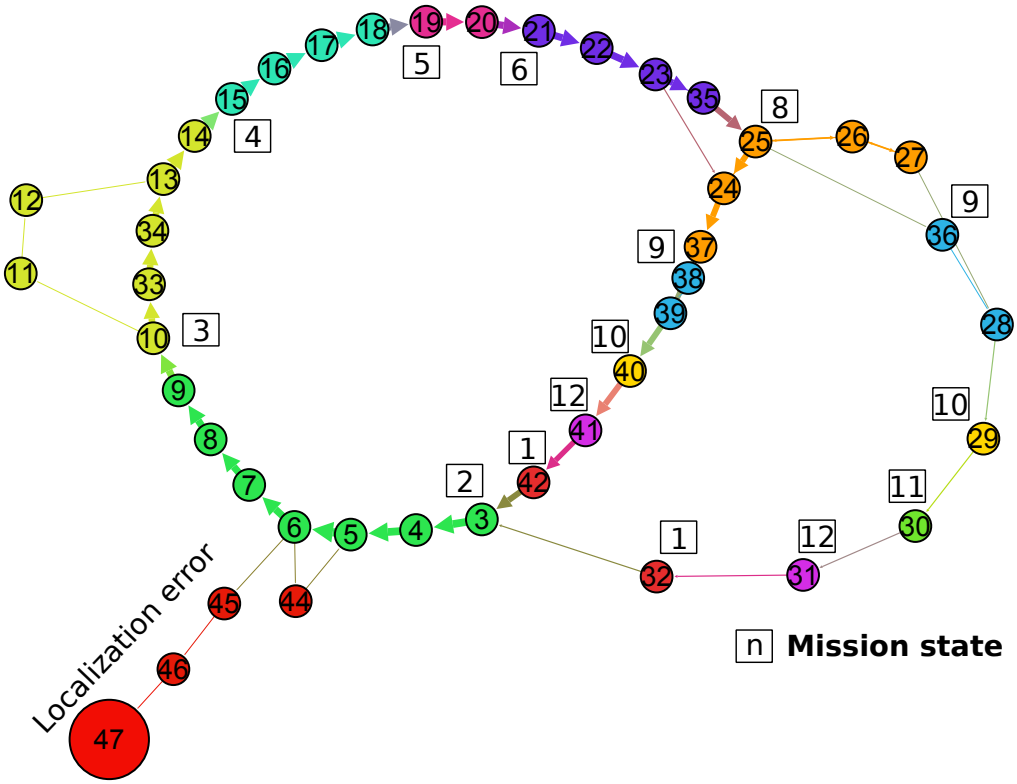

Figure 7.21: EMM model after running a force-based layout algorithm, adding node numbering and color coding of nodes according to m_{state}. The pattern of a loop through the states is now clearly identified and occasional outliers are not occluding the structure. The red nodes illustrate the localizer failure, which was clearly and timely detected.

From the situation model in figure 7.21 it can be concluded that the proposed level 1 situation model manages to capture characteristics from the AGV application data-set. The resulting model has clear structure, which matches the geographic position of events in the application as well as describing the sequential loop nature of the application.

Level 2: Comprehension of current situation

Although the structure of the EMM graph shows that a good perception of elements in the situation has been achieved, it is not certain that the states and spatio-temporal structure of the graph are useful for forming comprehension of the situation. As described by Endsley (1995): "*Level 2 SA goes beyond simply being aware of the elements that are present to include an understanding of these elements in light of pertinent operator goals*". In order to contribute to the comprehension of the current situation, states of the EMM model must represent a descriptive state for the system behavior. If such descriptive states have been achieved, e.g. rotating robot, attaching cart, etc., the comprehension of the current situation is accomplished by recognizing the current state within a certain temporal sequence.

Digging into the states of the generated EMM model, I initially investigate

the states in a critical situation for the robot application, the pick-up of the cart in $m_{state} = 3$. As figure 7.21 illustrates, the most likely sequence through this m_{state} is given by the sequence states s in X_{m3}:

$$X_{m3} = \{10, 33, 34, 13, 14\} \tag{7.13}$$

Or as illustrated in figure 7.22

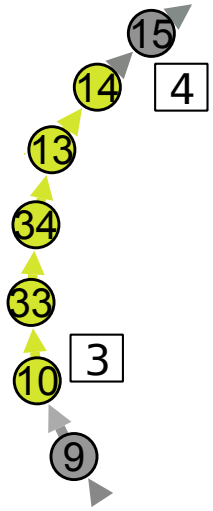

Figure 7.22: Sequence of states in $m_{state} = 3$ investigated.

Table 7.4 shows the center-values of the data-set of the states in X_3 except m_{state} which has been omitted as $m_{state} = 3$ for all nodes in X_s. The preceding state s_9 is also included in table 7.4, to illustrate that the state-change $s_9 \rightarrow s_{10}$ was caused only by change in m_{state}.

s_{id}	x_{map} [m]	y_{map} [m]	θ_{map} [rad]	$\Sigma_t(1,1)$ [m^2]	$\Sigma_t(2,2)$ [m^2]	$\Sigma_t(3,3)$ [rad^2]	i_{switch} [0/1]	**State change**
9	3.19	0.94	1.69	8.8e-5	1.1e-4	3.3e-4	0.0	Preceding state
10	3.19	0.94	1.69	7.2e-5	9.0e-5	1.9e-4	0.0	$m_{state} = 3$
33	**3.27**	**1.16**	**0.63**	7.0e-5	1.3e-4	3.8e-4	0.0	Moving and rotating
34	3.27	1.16	**-0.41**	3.6e-4	8.3e-5	6.4e-4	0.0	Rotation only
13	3.14	1.16	**-1.58**	8.8e-5	3.0e-4	6.4e-4	0.0	Rotation only
14	3.14	**1.42**	-1.58	3.0e-5	6.7e-4	5.2e-4	**1.0**	Move and attach cart

Table 7.4: Values of EMM states in X_{m3} to analyze the changes which formed the sequence. Significant changes in situation parameters are highlighted in bold text.

Table 7.4 shows the state sequence formed from docking to the cart in position 1 using the Kinect assisted docking script. The significant changes between states are described in the last column of the table, which are all easily interpretable

robot behaviors. The sequence well describes the actual sequence of the cart docking script. Table 7.5 compares the steps of the docking algorithm to the extracted states.

Dock state	**Dock behavior**	X_{m3} **EMM state**	**EMM behavior**
1	Find cart pose using front Kinect	**10**	State switch $m_{state=3}$
2	Move in front of cart	**33**	Move and rotate
3	Rotate 180°	**34,13**	Rotation only
4	Update cart pose using rear Kinect	**-**	Not detectable
5	Move until cart attaches	**14**	Move and attach cart

Table 7.5: States of cart docking script compared with the extracted states of the EMM model in the cart docking state at cart location 1 ($m_{state} = 3$).

As table 7.5 shows the automatically generated EMM model managed to capture all characteristics of the cart docking script from the raw values defined in the situation model. The only exception is the second cart pose estimation as this is not expressed in any of the variables of the situation model. So for the scenario of docking to cart in cart location 1, situation comprehension was accomplished.

As a second example of situation comprehension I examine the sequence from the beginning of movement from the pause location in state s_{25} where $m_{state} = 8$ and until the end of the loop sequence in state s_{42} and $m_{state} = 1$. This part of the model has been selected, as it is generated in the section of the AGV application where there was the largest variation in trajectory and divergence in state sequence. Following the sequence of maximum likelihood between the two points yields the following X_{end} set of states.

$$X_{end} = \{25, 24, 37, 38, 39, 40, 41, 42\} \tag{7.14}$$

Or as illustrated in figure 7.23.

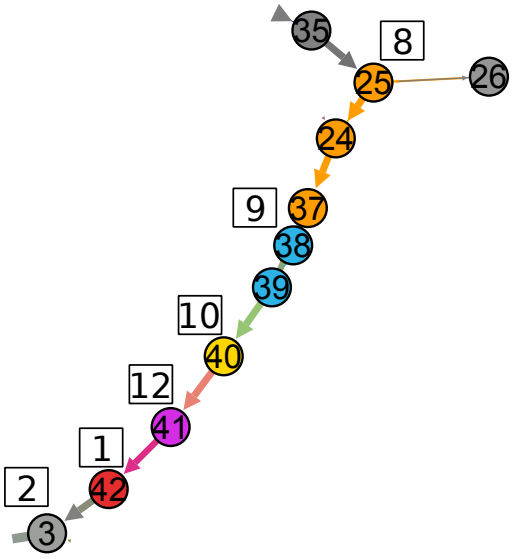

Figure 7.23: Sequence of states in maximum likelihood route from $s_{25} \rightarrow s_{42}$ investigated.

The center points of each state in the X_{end} data-set are listed in table 7.6 and again the preceding state s_{35} is included to observe the state change into the first examined state s_{25}.

s_{id}	x_{map} [m]	y_{map} [m]	θ_{map} [rad]	$\Sigma_t(1,1)$ [m^2]	$\Sigma_t(2,2)$ [m^2]	$\Sigma_t(3,3)$ [rad^2]	i_{switch} [0/1]	m_{state}	**State change**
35	1.79	0.12	3.45	1,3e-4	2.6e-4	3.9e-4	0.0	6	Preceding state
25	1.60	-0.05	3.32	2.1-4	4.2e-4	5.9e-4	0.0	**8**	State change
24	1.64	-0.04	**-2.77**	4.0e-4	4.3e-4	4.9e-4	0.0	8	Rotation
37	**1.39**	**0.19**	**-4.60**	8.8e-5	1.1e-4	3.6e-4	0.0	8	Move and rotate
38	**2.03**	**0.88**	**-3.58**	6.0e-5	4.7e-5	2.9e-4	0.0	**9**	Move and rotate
39	1.61	1.07	**-5.98**	1.8e-4	1.2e-4	6.5e-4	0.0	9	Rotate
40	**3.05**	1.09	-6.26	**3.3e-3**	1.9e-5	5.1e-4	**1.0**	**10**	Move with cart
41	3.24	1.44	**-7.86**	2.1e-5	8.0e-4	1.2e-4	**0.0**	**12**	Rotate and drop cart
42	**4.40**	**1.70**	**-5.92**	5.8e-5	2.9e-5	9.7e-5	0.0	**1**	Move and rotate

Table 7.6: Values of EMM states in X_{end} to analyze the changes which formed the maximum likelihood sequence from $s_{25} \rightarrow s_{42}$. Significant changes in situation parameters are highlighted in bold text.

As expected, the state change analysis in table 7.6 is not as clear and explicit as the earlier analysis in table 7.4. The sequence of cart pick-up in $m_{state} = 9$ is only represented in 2 states, compared to 4 in the X_3 sequence. The key question is, if the states are informative enough to allow comprehension of the current situation. Table 7.7 compares the behavior in the states of the X_{end} sequence to the goals of mission state m_{state}.

m_{state}	**Mission**	X_{end} **EMM state**	**EMM behavior**
8	Move to cart loc. 2	**25,24,37**	Rotate, then move while rotating
9	Pick-up cart	**38**	Move in position
9	Pick-up cart	**39**	Rotate 180°, pick cart
10	Move cart to loc. 1	**40**	Linear move
(11)12	Drop cart, move to start	**41**	Rotate 90°, drop cart
1	Arrive at start loc.	**42**	Move and rotate

Table 7.7: Mission goals of each m_{state} compared with changes in each states of the X_{end} sequence.

Comparing the resulting from situation comprehension in the EMM states of X_{end} with the behavior described by the m_{state} shows that each state in the X_{end} sequence is describing approximately the same behavior as defined in m_{state}. Some states, like s_{41}, indicate that the cart has been dropped although $m_{state} = 12$ instead of $m_{state} = 11$, which is the cart drop state. This is because the cart is dropped at the very end of $m_{state} = 11$ and the change in orientation as well as i_{swicth} and finally $m_{state} = 12$ triggers the transition $s_{40} \rightarrow s_{41}$.

In summary, the state sequence from $s_{25} \rightarrow s_{42}$ does form a fairly good comprehension of the situation, as the interpretation of the X_{end} sequence in table 7.7 would serve as just as reasonable an explanation of the situation in the sequence as the mission description of the m_{state} columns.

Although the situation model was not designed specifically to capture localization errors, the failure in the last loop of the experiment was captured in four error states $s_{44} - s_{47}$ in figure 7.24. Despite that the robot only left its planned trajectory slightly, the dramatic increase of Σ_t was assessed as a new situation. Node s_{44} captured a rising anomaly but returned to the normal state sequence for one transition before the anomaly grew enough for the sequence to split completely from the regular sequence in nodes $s_{45} \rightarrow s_{47}$.

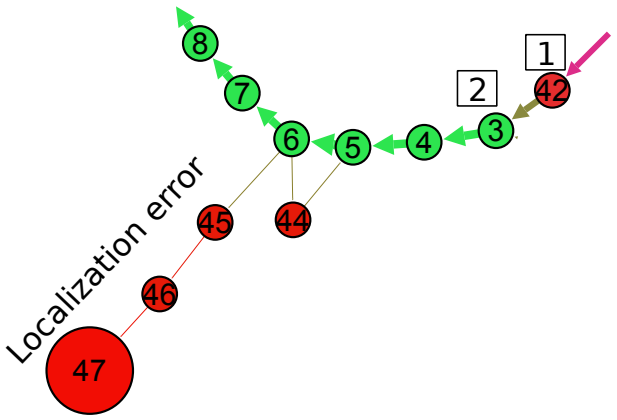

Figure 7.24: The situation of localization failure was successfully captured by the EMM by a number of states diverging from the known states.

This in-depth analysis of significance of states in the generated EMM models compared with expected robot behavior leads to the conclusion that a good level situation comprehension was achieved by the situation assessment process. In the clean section of the data-set, all significant information was successfully extracted from the docking procedures at cart location 1 in the state-set X_3. Even in the slightly noisier section of the data-set a fairly long sequence of states in the set X_{end} proved informative and able to describe the situation in good accuracy and most importantly by correctly identified events.

Level 3: Projection of future status

When thinking of mobile robotics control as purely reactive systems, it would be sufficient to implement the situation comprehension, in order to react to certain situation. Prediction of future events has been described as the "ace-factor" for fighter-pilots, where correct prediction of what was going to happen gave a winning edge compared to pilots without that ability. The same applies to mobile robots, where prediction and avoidance of errors and critical situations are much better than good error recovery, as errors always degrade performance and sometimes recovery is even impossible.

Projection or prediction of future status in situation has always proven slightly difficult. For humans Endsley (1995) describes the achievement of Level 3 SA as "*it is achieved through knowledge and dynamics of the elements and the comprehension of the situation (both Level 1 and Level 2 SA)*". In the spatio-temporal EMM model of situations prediction of future state-transitions is generated by maintaining transition likelihoods along edges between the node structure (Level 1 and 2) of the EMM model. These transition likelihoods represent the statistical likelihood that transition from one state to another is going to occur, based on the transition patterns identified, when the model was constructed.

The prediction engine (presented in section 4.5 on page 71) continuously maintains maximum transition likelihoods between the current state of the online EMM model to all other states in the model. This way, watches can be set up to monitor maximum transition likelihoods towards selected critical states in order to activate preventive measures if the likelihood if a critical situation rises above selected thresholds.

In the following experiment, I investigate the prediction capabilities of the produced EMM model and how prediction of state transitions could be used to possibly improve the reliability of the AGV Application. State transition prediction can be considered especially useful for prediction of events in the very near future, as reliability might be highest, but for well-known models prediction with longer time frames might be accomplished as well.

The first state-sequences to look into is obviously the X_3 and X_{end} sequences. Running the prediction calculation on the maximum likelihood of the X_3 sequence transition from $s_{10} \rightarrow s_{14}$ yields a maximum likelihood

$$L(s_{14}|s_{10}) = \prod_{i \in X_3} a_{i,i-1} = 0.917 \tag{7.15}$$

which is a fairly good likelihood for the robot to follow the X_3 sequence of states if it is recognized to enter s_{10}. Inspecting the sequence in 7.19 where the value at each s_n node represents the cumulative likelihood calculated by the chain rule and values at each $\rightarrow$-transition are the individual edge transition likelihoods.

$$s_{10}(1.00) \xrightarrow{0.917} s_{33}(0.917) \xrightarrow{1.0} s_{34}(0.917) \xrightarrow{1.0} s_{13}(0.917) \xrightarrow{1.0} s_{14}(0.917) \quad (7.16)$$

Although the $L(s_{14}|s_{10})$ is fairly good for the X_3 sequence, the analysis in 7.19 reveals that the critical transition in the sequence is from s_{10} with a $1 - 0.917 = 0.083$ likelihood to follow the edge towards s_{11} instead of s_{33}. If s_{11} was known to be problematic, a recovery strategy could be prepared or mission control could be adjusted to reduce this transition likelihood even further.

The challenge of longer term prediction becomes evident when analyzing the X_{end} sequence from $s_{25} \rightarrow s_{42}$. Maximum likelihood calculation from the prediction engine yields

$$L(s_{42}|s_{25}) = \prod_{i \in X_{end}} a_{i,i-1} = 0.141 \quad (7.17)$$

Despite that the sequence has the maximum likelihood for transition between s_{25} and s_{42}, it contains so many possible branches that the cumulative likelihood becomes quite low and difficult to use for any prediction. Inspection into the chain of transitions can provide further details on the critical sections and how this sequence can be applied for prediction.

$$\begin{aligned} s_{25}(1.00) \xrightarrow{0.71} s_{24}(0.71) \xrightarrow{0.83} s_{37}(0.59) \xrightarrow{1.0} s_{38}(0.59) \xrightarrow{0.5} s_{39}(0.30) \xrightarrow{0.53} \\ s_{40}(0.16) \xrightarrow{1.0} s_{41}(0.16) \xrightarrow{0.90} s_{42}(0.14) \end{aligned}$$

Already at the transition from $s_{25} \rightarrow s_{24}$ the likelihood drops $L(s_{24}|s_{25}) = 0.71$, as possible branches are also towards s_{26} with $L(s_{26}|s_{25}) = 0.21$ and towards s_{36} with $L(s_{36}|s_{25}) = 0.071$. These likelihoods give a fairly good estimate on which transitions to prepare for next, thus in this case, it would most likely be to s_{24} but possible transitions to s_{26} and even s_{36} must also be handled in the controller. If a transition towards s_{26} occurs, the most likely sequence through the application loop changes to X_{end2}

$$X_{end2} = \{s_{26}, s_{27}, s_{28}, s_{29}, s_{30}, s_{31}, s_{32}\} \quad (7.18)$$

with the cumulative transition likelihoods

$$L(s_{32}|s_{26}) = 0.20 \qquad L(s_{42}|s_{26}) \qquad = 0.056$$

So if different control strategies are needed for following the sequence from $s_{26} \rightarrow s_{32}$ than $s_{25} \rightarrow s_{42}$ (it is not in this case), then the strategy should already be prepared in s_{25} and evaluated if the controller is capable of executing the sequence in X_{end2}. If not, countermeasures should be activated to ensure that the robot would stay on X_{end}.

Using the localization error as regular section of the data-set sequences would not be justified as there is only this one instance of the error. The localization error can be considered as an example of an unexpected random error or black swan situation. As no prior evidence could point towards this error happening, it could not be predicted by the EMM.

Although no planning could be executed in advance of the error, it can be used as a postmortem examination of the likelihood of a random error occurring in the AGV application and thus evaluate its reliability. Investigating the maximum likelihood for $s_6 \rightarrow s_{47}$ yields a likelihood for random "black swan" events of

$$s_6(1.00) \xrightarrow{0.071} s_{45}(0.071) \xrightarrow{1.0} s_{46}(0.071) \xrightarrow{1.0} s_{47}(0.071) \tag{7.19}$$

which corresponds with a failure in the beginning of the 14th loop ($\frac{1}{14} = 0.071$) and denotes the reliability of the AGV application at the current state of the EMM model. Running the model for many additional loops would naturally improve this prediction rate and possibly even create a certain pattern in the errors, which would be useful for prediction and setting up control rules to avoid the errors.

This prediction analysis shows that transition likelihoods within the EMM model can be very useful for predicting future status of the modeled application. In regions where there is clear graph structure without many branches, good prediction results can be obtained several states into the future. Where the graph-structure contains many branches, it can be useless to perform rely on maximum likelihood calculations through many state transitions but prediction into the very near future can be useful to prepare for branches in the graph and possibly improve mission control. Reliability of the application can be estimated by the likelihood a new branch of states would be assigned which could be caused by a random "black swan" error event, or simply a new structure in the graph, caused e.g. by environment changes.

7.5 Discussion

The presented results are primarily obtained in context of navigation tasks for mobile robots, but the capabilities of the proposed situation assessment framework should cope with a variety of real world application scenarios. The experiments were selected with the purpose to highlight the capabilities of the proposed situation assessment framework and address the different strategies presented, such as pre-processing of data to extract features, error detection on specifically selected data-sources and general situation assessment using a heterogeneous situation model.

Characteristic of the selected experimental scenarios has been that the input data have in most cases been floating-point numeric data, such as laserscanner distances, robot poses and localizer covariance. This has been chosen deliberately as this is how most information is represented in the control-oriented architecture of most mobile robot frameworks. The experimental results demonstrated that proper higher level knowledge can be extracted from the actual data that are used in the controller without producing artificial symbolic facts or using over-simplified models.

Situation assessment has demonstrated its benefit in a number of different applications. The small module which continuously monitors the approach of narrow passages is an example of small house-keeping situation assessment modules which work without being tied too closely to the robot application itself. Development of a number of these situation assessment modules for various purposes will supply the robot mission management with situation comprehensions to adjust the control strategies and to sanity check the mission execution. The situation model of the large-scale experiment includes a large number of heterogeneous data to form a complete situation assessment over a particular part of a mobile robot mission. This type of situation model learns the robot behavior while performing repetitive tasks and thus enables detection of deviation from the normal sequences as well as the recognition of the overall status of the situation. It efficiently combines the internal robot status and status of the environment. Finally, this model proved useful for prediction of future status and preparing the robot for approaching state changes and branches in course of future actions. The prediction framework proved to depend significantly on the model characteristics at the point of prediction, where some regions allowed high likelihood prediction for a fair number of future state transitions and others only provided a low quality prediction for the very next state transition.

Although the context of this thesis and all examples in this chapter have been within the domain of mobile robotics, nothing in particular has been designed in the architecture to only suit this domain. Actually, the resulting situation assessment framework might suit other domains even better, as dealing with the large level of uncertainty that exists in the domain of mobile robotics is one of the key challenges for situation assessment framework. Industrial automation and industrial robotics operate in much more isolated environments and with a much larger count of identical cycles, which are an ideal environment for situation assessment. Thus the future work on the proposed situation assessment platform will continue within industrial robotics for flexible automation for small and medium sized companies.

7.6 Performance considerations

Performance is a key issue, when designing systems to deal with data streaming data sources. Clustering data-sets extracted from high volume data sources such as robotic vision systems or laserscanners are challenging both in terms of dimensionality, data rate and the number of clusters recognized in the data-set. When learning structures from these data-sets in an on-line way, the number of detected clusters can become very large, containing many interesting temporal patterns in the same structure. Combining the large data-sets with data rates of 100 Hz (10 ms) for robot control, 75 Hz (13.3 ms) for laserscanners and 30 Hz (33.3 ms) for vision systems, it is necessary to look into high performance computing to design a system which can handle this load.

A recent trend of achieving High Performance Computing (HPC) capabilities without investing fortunes in refrigerator sized computing clusters is the use of General Purpose GPU computing, using the massive parallel computing capabilities of common Graphics Processing Units (GPUs). Getting a position on the Top 500 list for the world's supercomputers requires calculation performance of 209 TFLOPS[17], where a high-end consumer GPU like the AMD Radeon HD7970 can produce as much as 4.3 TFLOPS for less than 400 euro.

GPGPU technology is also an optimal solution with respect to mobile robots. Low power platforms for mobile computing do now include potent GPUs, where platforms like the AMD E-450 computing platform include the Radeon HD6320 GPU, capable of producing 80 GFLOPS single precision with a power consumption of less than 18 W. In comparison an Intel quad core i7 920 CPU at 2.6 GHz can deliver 69 GFLOPS single precision consuming 130 W. Moderately sized computing platforms like the E-450 are ideal for medium sized mobile robots, where power consumption is critical, but yet demanding data processing from cameras and laserscanners must be handled.

With this goal in mind, a GPGPU version was designed of the Jaccard similarity coefficient for dynamic stream clustering. Key design decisions were:

- The algorithm should be portable across GPU vendors, so OpenCL was chosen as implementation platform.
- Modest and mobile suitable hardware specifications must be assumed.

The development and test platform was a modest Intel quad-core i7 720QM at 1.6 GHz (with Turbo Boost to 2.8 GHz) and an Nvidia Quattro 880M GPU featuring 48 GPU cores at 550Mhz (the E-450 has 80 cores at 600 Mhz).

7.6.1 Algorithm analysis

As previously described, the Jaccard similarity index is given by

$$f(A,B) = \frac{A \cdot B}{|A|^2 + |B|^2 - A \cdot B} \tag{7.20}$$

[17] Top500 Supercomputer list, June 2012, http://www.top500.org/list/2012/06/100

For an efficient calculation of the index three vector products are required:

- $A \cdot B$
- $|A|^2$
- $|B|^2$

These calculations themselves leave good possibility for parallization, as the vector products can be done as a parallel multiplication of each vector element, followed by a reduction sum of the multiplication products.

Furthermore, the vector B represents one data-vector of the cluster of existing states, where the cluster is a growing collection of state vectors in the counts from tens to tens of thousands of vectors. So each similarity calculation can be parallelized by the number of vectors.

The square product $|A|^2$ of vector A is calculated on basis of the new vector and is only required as a scalar in the Jaccard calculation between all B vectors in the cluster. That makes it obvious to perform $|A|^2$ initially on CPU and only parallelize $A \cdot B$ and $|B|^2$.

7.6.2 Memory management

Many aspects of this algorithm are heavily related to memory operations and allocation. Memory management is a general problem in GPGPU as the simple GPU cores do not have features like branch prediction and automatic caching. Main memory access is enormously slow compared to the calculation capability of the GPU, so much performance is gained by organizing memory layouts so that GPU cores can retrieve blocks of memory to process in a number of threads instead of individual assess. As I will return to later, the algorithm itself is strongly memory-bound, but there is also a general challenge of efficient memory management of an ever growing list of vectors.

The memory management is implemented using large sequential memory area, where individual vectors are placed sequentially and simply addressed by a running index. Although being slightly messy in the addressing, this implementation allows efficient whole-block transfers to the GPU and efficient re-allocation for growing buffers.

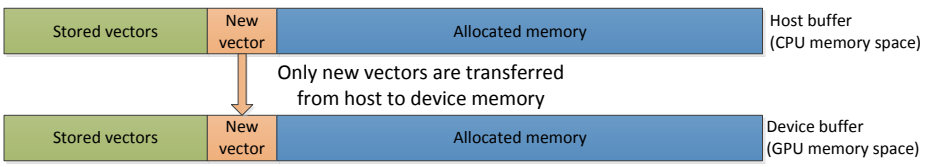

Figure 7.25: The GPU device memory holds a persistent mirror of existing vectors in the cluster, so only new vectors must be transferred to the device before running a calculation.

7.6.3 Measurement of performance

To evaluate the performance of the algorithm the runtime of the calculations must be compared with the number of calculations. As previously mentioned, a vector product can be described as a product of each element and the sum as in the pseudo code snippet in listing 7.2.

```
for each i in d :
  product = A[i] * B[i]
  productSum += product
```

Listing 7.2: Example product of vector `A` and vector `B` with dimension `d`

The number of floating point operations in listing 7.2 are one multiplication and one addition resulting in $2 \cdot d$ operations for each vector. Extending the calculation to calculating the Jaccard similarity index between vector A and B with the dimensions d against all n elements in the cluster, the number of floating point operations can be calculated from the pseudo code in listing

```
sA = |A|^2                          //2 * d FLOP
for n {                             //n * (
  sB = |B|^2                        //  2 * d FLOP
  pAb = A * B                       //  2 * d FLOP
  jaccard = pAb / (sA + sB - pAB)   //  3 FLOP
}                                   //)
```

Listing 7.3: Pseudo code of the calculation steps for running the jaccard similarity index on between all elements of the cluster.

Resulting in a total number of floating point operations pr. jaccard calculation of

$$N_{FLOP} = 2 \cdot d + n \cdot (2 \cdot d + 2 \cdot d + 3) = 2 \cdot d + n \cdot (4 \cdot d + 3)\text{FLOP} \tag{7.21}$$

Running a naive CPU implementation of the pseudo code for a vector dimension $d = (2, 32, 512)$ elements yields the following performance

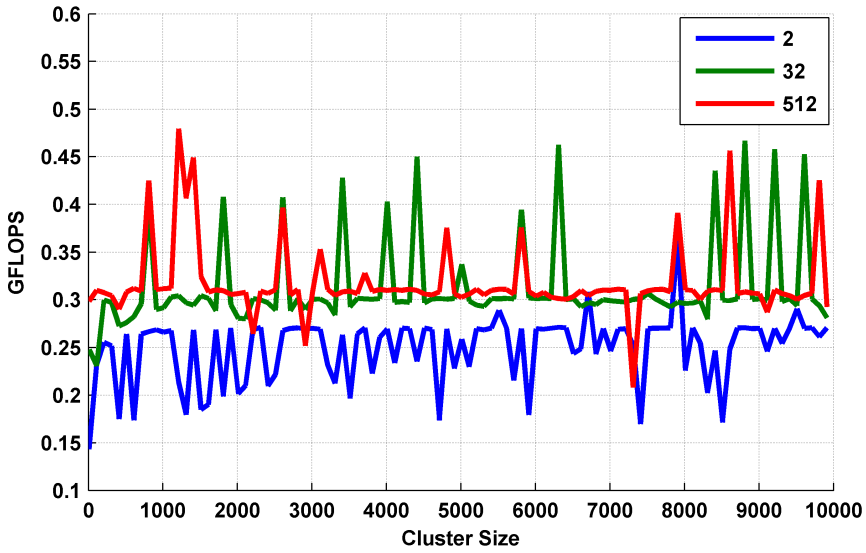

Figure 7.26: Base CPU performance of calculating the Jaccard similarity index with vectors of dimension $d = (2, 32, 512)$ and cluster elements in the range from $n = [1, 10000]$.

7.6.4 GPU implementation

GPU cores are simple arithmetic units whose performance and capability itself are fairly low but their true power lies in their massive number. The architecture of most modern GPU is built with a number of multiprocessors which orchestrate a number of stream processors or cores (often 32 pr. multiprocessor). Each multiprocessor has one single instruction unit, forcing all cores to execute the exact same sequence of instructions simultaneously, only separated by a number of indices which allow them to select different elements in memory to process. Depending on the architecture, the GPU can have from 1 to more than hundred stream processors to handle different (or identical) instruction sequences.

Within OpenCL (and also Nvidia CUDA), the threading model can be design in a grid of $m \times n$ threads, all executing the same kernel. The cores in each multiprocessor executes m threads and the n thread blocks are allocated between the multiprocessors, as they become idle. For the Jaccard similarity calculations this model is very suitable resulting in a dxn threading model as illustrated in figure 7.27.

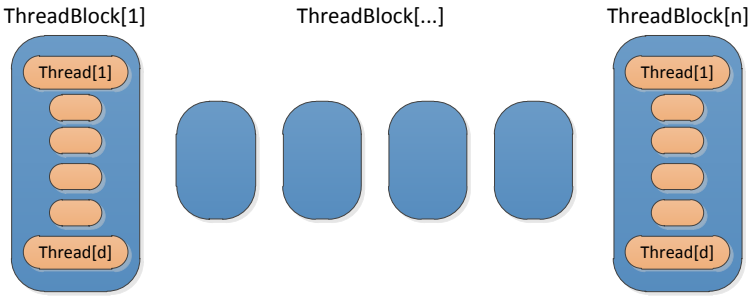

Figure 7.27: GPU Threading model for the Jaccard similarity calculations. A matrix structure of $d \times m$ threads unrolls the two loops required for the vector product and cluster element iterations.

This breakdown allows also using shared memory for operations within one thread block, giving a significant speedup on multiple accesses to the same data. The addressing strategy even ensures a fully coalesced reading of the memory. Unfortunately, the decomposition also introduces some problems. Multiplication of the individual elements works excellently in this model, but after the multiplications the vector elements must be summed in a reduction operation to one final scalar which is not evident to run in parallel.

The optimization solution is to perform a parallel reduction of the final product. The parallel reduction does not ensure full utilization of the GPU cores, but it is significantly more efficient than letting a single core perform all the summarization. The principle of the model is illustrated in figure 7.28.

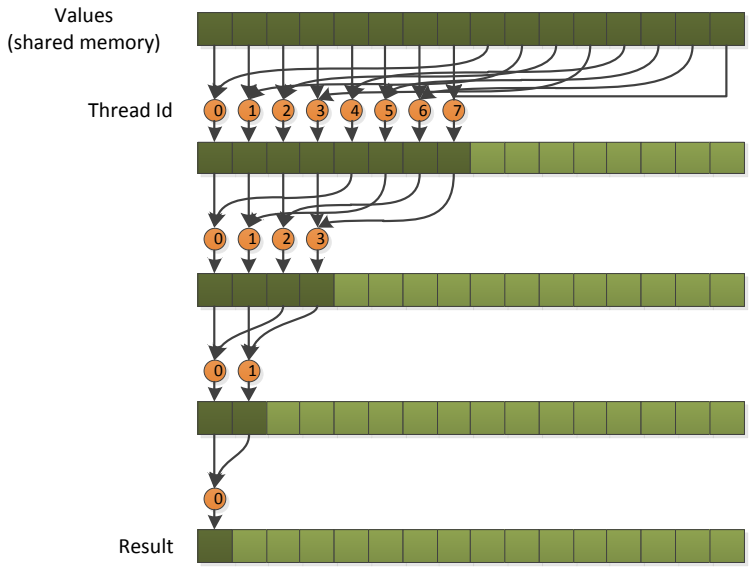

Figure 7.28: Summarization of elements from a vector using parallel reduction.

The calculation of the square of the new data vector $|A|^2$ is still calculated on CPU and passed into the GPU kernel as a scalar function parameter. Running the calculation with vector dimensions of $d = (2, 8, 32, 128, 512)$ on the Nvidia Quattro 880M 48 core GPU yields the performance as seen in figure 7.29

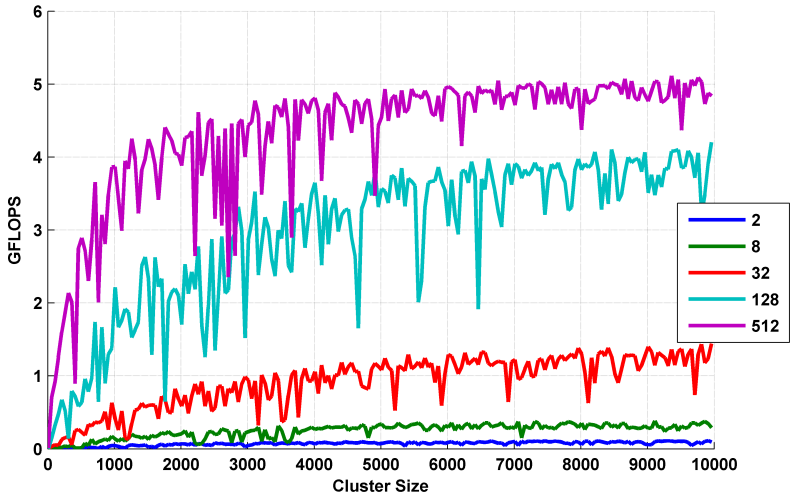

Figure 7.29: Performance of the optimized GPU implementation of the Jaccard similarity index with vector dimensions of $d = (2, 8, 32, 128, 512)$ and cluster elements in the range from $n = [1, 10000]$.

7.6.5 Performance comparison and evaluation

The GPU implementation was done in two variants; one was a simple port of the naive CPU version where each complete vector calculation was allocated to one GPU core. The optimized version uses the parallel reduction technique as well as a few simple Nvidia specific optimizations. The performance measures can be seen in figure 7.30 and in table 7.8.

Throughout the implementation, even the initial version of the GPU algorithm outperformed the CPU version of the algorithm. Table 7.8 illustrates the performance achieved between the implementations. On smaller vector dimensions the GPU version suffers some performance impact, as the setup and data transfer times become significant. The table also illustrates cumulative performance enhancement compared to the CPU implementation.

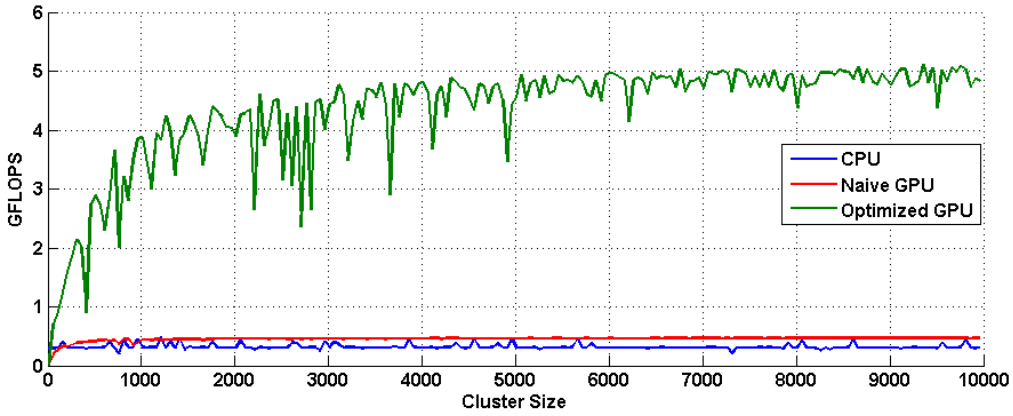

Figure 7.30: Performance comparison between the CPU and different GPU implementations on a vector dimension of 512 elements and cluster elements in the range from $n = [1, 10000]$.

Implementation	**1x2 vector** [GFLOPS]	**1x32 vector** [GFLOPS]	**1x128 vector** [GFLOPS]	**1x512 vector** [GFLOPS]
Naive CPU	0.272	0.316	0.332	0.316
Naive GPU	0.194 (0.71x)	0.429 (1.35x)	0.464 (1.39x)	0.469 (1.48x)
Optimized GPU	0.095 (0.35x)	1.253 (3.96x)	3.775 (11.37x)	4.947 (15.65x)

Table 7.8: Performance comparison of CPU, Unoptimized GPU and optimized GPU performance on vector dimensions of $d = (2, 32, 128, 512)$. The performance measure is average performance of processing a cluster with element counts from $n = 1, 10000$.

Performance measures from table 7.8 clearly show the tremendous benefit of using a GPU for performing the Jaccard calculations. Using the optimized GPU variant a performance enhancement of more than 15 times was achieved on large vector dimensions. Low power systems with moderately performing GPUs are a cost effective way to enhance performance in critical and calculation intensive sections. It is evident that there is considerably more effort invested in making a proper GPU implementation than just using the CPU but more and more robot perception systems have a number of intensive algorithms running, such as stereo vision, object recognition and point cloud processing.

Considering the sample-time requirements of real-time robot systems, another issue worth noting is the worst-case execution time. If we consider the clustering operation on a data-vector with a dimension of 512 elements and a cluster size of 10.000 vectors, the execution time can be seen in table 7.9.

Implementation	Execution time
Naive CPU	85.87 ms
Naive GPU	58.02 ms
Optimized GPU	5.48 ms

Table 7.9: Worst case execution time for similarity calculations of a vector with dimension $d = 512$ against a cluster with $n = 10000$ elements.

Table 7.9 shows where the real benefit of using GPU accelerated clustering becomes significant. Using the GPU it is possible to follow the data rates of high-rate components such as the robot controller and laserscanners (10ms and 13.3 ms respectively) which would not have been possible using the CPU version.

Using OpenCL it is fairly simple to design software to only run the GPU kernels when specific GPU hardware is available or data-sets match. Kernels are integrated into traditional programs using the standard compilation framework and by invoking the OpenCL library to Just-In-Time compile the kernels to the available hardware, which could be GPUs from Nvidia, AMD, Intel or CPUs from Intel, IBM or Apple, or in the future even FPGAs. This flexible approach also allows simple integration into many programming languages, such as Python and Java.

7.6.6 Summary

Although the on-line streaming clustering engine has been designed for high throughput and on-line processing, data-volumes of modern robotic systems can become quite intense. Using GPGPU to run the distance calculations for clustering in parallel on a graphics processor (GPU), it was demonstrated that processing performance could be improved up to 15 times on hardware suitable for mobile platforms. Using the GPU implementation processing times can be reduced to a level, where the clustering can follow high-rate sensor systems, even in high data dimensionality and high numbers of clusters without saturation. A further key benefit of using the GPU is that much of the workload is shifted from the robot system CPU, relieving it to other tasks, such as controlling the robot.

8

Conclusions

The use of mobile robots in the global scene, both in industrial and domestic settings is rapidly expanding. From being restricted to high volume logistics in factory settings and vacuum cleaning in our homes, mobile robots are now being introduced in hospitals, in agricultural fields, as packaging assistance in the mail order industry, and have even gained the license to drive autonomously in the streets of the US state Nevada under supervised test conditions.

An enormous body of scientific results within advanced navigation and perception exists and some of these results are slowly finding their way into commercial products, such as the Neato XV-15 vacuuming robot which uses SLAM. Using these technologies enable a significantly higher degree of freedom in the capabilities and application for the robots, e.g. from being confined to navigate in areas with reflective markers for laser guidance to moving freely in offices using only natural environments for navigation. This new freedom and flexibility comes at a price of higher complexity and loss of reliability.

A key obstacle for many new robot applications of finding their way into the mass markets is the reliability of the applications. Operation in changing and only partially known environments requires a perfect recognition of the environments as well as the capability of understanding what is going on in order to take the right decisions and continue operation, even when critical situations occur.

The work presented in this dissertation targets the challenge of understanding the situation of the mobile robots and proposes a strategy towards answering the question of what is going on? As well as the equally critical question of what is going to happen next?

With this knowledge, the mobile robots will have the potential to handle critical situations before they develop into failures and improve the reliability of applications in a number of new domains.

8.1 Summary of Achievements

The overall contribution of the presented work is an introduction of the methodology and presentation of a framework for situation assessment in the domain of mobile robotics. The focus has been on creating connection between the challenges of mobile robot applications and the scientific approaches to situation assessment. A further focus has been on how to interface situation assessment with real-world robot platforms to demonstrate and verify the achieved results as well as enabling the integration of situation assessment into future mobile robot applications.

In summary, three major contributions have been presented:

Situation Assessment.
Analysis and synthesis of a model for situation assessment in the domain of mobile robots.

Modeling framework.
A novel situation assessment modeling framework for on-line construction, comprehension and prediction of situations.

Software architecture.
A new software architecture to connect robotics software frameworks and situation assessment.

The following sections outline the details of the achieved results.

8.1.1 Situation Assessment for mobile robots

The work presented in this thesis is the first of its kind to directly connect situation assessment from an information fusion perspective to the challenges of on-line and real-time applications of mobile robotics with the many levels of uncertainty and information sources that are present in this domain.

To support this contribution a thorough analysis is presented of the use-cases for mobile robots in commercial use, both industrially and in the service industry, together with state-of-the-art applications within research robotics. To investigate the connection between these applications and higher level processing like situation assessment, an analysis of community driven software frameworks used for development of these state-of-the-art applications was presented. Common for all analyzed frameworks is, that an enormous volume of work has been invested in modules for low level control and environment perception, which all community frameworks offer in abundance. However, analysis shows that when it comes to application development, hierarchical control architectures and reliable mission control, none of the frameworks offers more than very rudimentary features and tools. As result, applications are executed in these frameworks without any plan B or back-up strategies if modules should fail, or if any of the basic assumptions the control or even the entire application relies on, should fail.

Furthermore, a discussion and definition of situations in general were presented and the concept of critical situations for mobile robots was addressed.

Critical situations for the mobile robots are in particular when the basic assumptions of application design fail. These assumptions will often not be detected or even be directly detectable for the controller, e.g. motor failures or low level communication errors. However, these errors could be directly detectable but are often simply ignored by current state-of-the-art algorithms, e.g. errors in the localization system.

As a result of this analysis I propose a technical model definition of situations as spatio-temporal constructs of events and furthermore propose three operational strategies for achieving situation assessment in the spatio-temporal patterns:

Type 1: Detection of known spatio-temporal relations.

Type 2: Deviation from known spatio-temporal patterns.

Type 3: Detection of known critical situations.

A fourth type was presented concerning "Black Swan events" which are unpredictable events with major impact. This type is not covered in the work of this thesis in larger detail than when the events fall into the category of events covered by the type 2 strategy.

8.1.2 Situation Assessment modeling framework

A novel situation assessment modeling framework has been presented which has the capabilities of specifying and representing situations in a general technical context. The analysis and synthesis of the modeling framework has been with a mobile robotics context, in which it has been evaluated with good results.

The widely adapted model of human situation awareness by Endsley (1995) suggests that human situational awareness can be organized in three levels:

Level 1: Perception of the elements in the current situation.

Level 2: Comprehension of the current situation.

Level 3: Prediction of future status.

The goal of the situation model design was to represent these three layers in the model for best possible interpretability by humans as well as making it easier to apply common sense when parameterizing the models.

Perception of the elements of the situation is parameterized in the XML based situation model, which is part of the proposed Situation Assessment Platform architecture. Through this model, data from heterogeneous data sources can be fused to form a comprehensive set of elements to capture the characteristics of a scenario under investigation.

Comprehension of the situation is achieved by capturing the characteristics of the situation in a spatio-temporal structure. The spatio-temporal structure is

represented in the framework by applying the Extensible Markov Model (EMM) by Dunham et al. (2004) as fundamental model. The EMM is a dynamic first order Markov model, which has the capabilities of evolving through time while maintaining its structure of state visits and transition likelihoods. This modeling approach allows the situation model to be developed under on-line execution of an application scenario for the robot and represent the situations as the robot actually perceives them. Furthermore, the model can evolve and adjust as the environment, the application, or even the robot changes through time.

On-line streaming data retrieved from robotics frameworks are connected to the EMM by using stream-based clustering, which is ideal as a non-model driven approach to handling sensor noise and at the same time performing on-line data reduction. In most cases, it is only desired to capture the overall spatio-temporal dynamic structure of the data in the EMM, and thus noise and minor variations should be suppressed.

Prediction of future status is achieved by a proposed prediction engine which combines the use of the chain rule of probability and a modified version of Dijkstra's algorithm for on-line calculation of maximum likelihoods sequences in the EMM. As the EMM continues to evolve through on-line execution, likelihoods of transitions changes and critical states might appear in the model which would be beneficial to monitor in real-time. The prediction engine continuously calculates the maximum likelihood from the current state to any other state in the EMM to predict the likelihood of ending in an identified critical situation and thus enable the controller to activate counter measures if necessary.

8.1.3 Software architecture

A new software architecture is proposed to integrate the contributed software modules in a flexible, expandable, and modular framework. The architecture is implemented in the Situation Assessment Platform (SA-Platform) software package, which offers a cross platform and GUI driven platform for easy interaction and configuration of the system.

A key contribution of the software architecture is the flexible dataSourceService module, which enables seamless interfacing to the heterogeneous world of data sources used within robotics. Data sources are implemented in completely independent modules, and data are distributed in the architecture using the publish-subscribe principle and event driven call-backs.

The evolution of the spatio-temporal EMM can be followed in real-time in tables of the SA-Platform, but for true visualization of the graph-based structure, the development and current status of the EMM are streamed on-line to the Gephi graph visualization software. Using Gephi the layout of the graph-structures can be optimized in real-time, and past states together with predicted future state transitions can be highlighted using color coding. Furthermore, specific details from the investigated data-sets can be channeled to the visualization framework, such as geographic location of nodes in the layout or states of mission execution for color coding of nodes. The use of on-line Graph visualization is a tremendous benefit for transferring comprehension of situation models to

operators.

The use of this software framework has enabled demonstration and evaluation of the situation modeling framework using on-line and real-world examples. This enabled the testing of the capabilities and performance of the situation assessment, not only on recorded and pre-processed data-sets but under real mobile robot applications, and thus investigates how the system performs under real-time conditions.

8.2 Experimental results

Experiments were conducted to evaluate the situation assessment framework in three scenarios, where each of these experiments was designed to evaluate a certain type of situation assessment strategy:

- Narrow passage detection (Type 3)
- Detection of errors in AGV localization tracking (Type 2)
- AGV Situation characterization and prediction (Type 1)

The narrow passage detection experiment showed that feature extraction from raw laserscanner data could be used with situation assessment to detect and predict when the robot approached narrow passages. Narrow passages are notoriously known as critical situations for mobile robot navigation systems. The result shows that situation assessment can be applied to this purpose successfully.

The type 2 situation assessment strategy, deviation from known patterns, was investigated in the detection of errors in AGV localization tracking experiment. In the experiment the relation between AGV position and localization covariance was investigated to perform early detection of navigation errors before critical failures occurs. The result showed that this strategy could successfully be applied to detect a localization error on an AGV logistics robot, which performed a repetitive motion cycle. In the third loop, the AGV localizer failed and this was rapidly detected as an anomaly in the regular pattern of robot location and localizer covariance.

In the AGV Situation characterization and prediction experiment, a 25 minute long sequence of an AGV performing 13 loops through a sequence of moving a cart between two locations, was used to investigate the type 1 strategy, detection of known spatio-temporal relations as well as prediction performance. The large sequence was successfully applied to achieve all of Endsley's three levels of situation awareness, as it was demonstrated that the produced EMM provided a good perception of the elements of the situation, as well as the state patterns in the model provided a good comprehension of the current situations. On basis of large number of state transitions embedded in the model, the EMM also proved useful for extracting likelihoods for prediction of future status for the robot.

8.3 Future work

Although the work of this thesis presented successful results, a number of open problems were left for future research.

The strategy chosen for clustering streaming data uses a fairly simple principle for assigning correspondences between new data-sets and existing clusters. The chosen approach is sensitive to the choice of similarity threshold as well as both the absolute scale and offset of processed data. Investigation in more advanced and capable clustering strategies will improve the representation quality of the EMM and make design of proper situation models simpler.

In order to significantly improve the performance of mobile robot systems, it is not enough to have a system for comprehension of the situations. A component for on-line inference is required to link the situation assessments to actions in the robot control system. Inference is often discussed in relation to development of robotic systems with cognitive capabilities, and sometimes even mentioned as "just inference". Although inference seems fairly simple to us humans, the inference problem is one of the key challenges within cognitive robotics, where current state-of-the-art still required complex structures of ontologies and knowledge representation before good inference results are possible.

Although mobile robotics has been the background of this work, there is nothing in the framework which limits it to this domain. Some other domains, for example industrial robotics might prove even more suitable, as the variations in the processes and motion patterns are significantly less than for mobile robots. As a result of this, the use of situation assessment in this context will be investigated in the EU FP7 project SMErobotics under the ICT challenge 2 call and in collaboration with a strong group of partners as illustrated in figure 8.1.

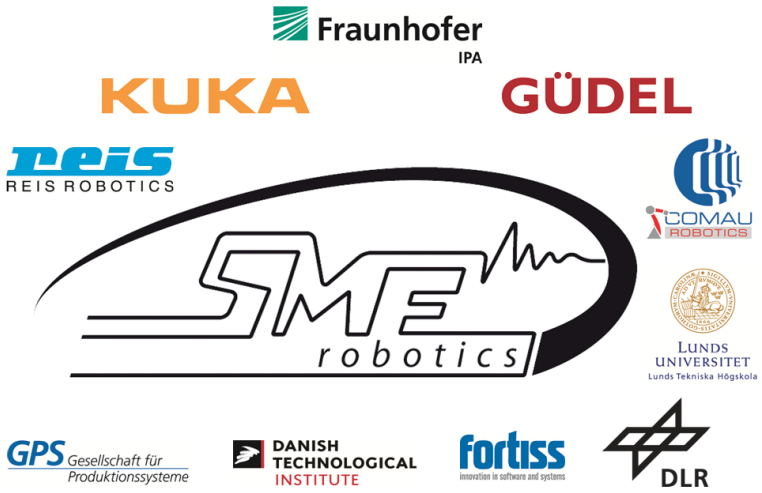

Figure 8.1: The use of situation assessment in a human-robot cooperation scenario using industrial robots is investigated in the EU FP7 project SMErobotics.

In SMErobotics situation assessment will be applied in a human and robot cooperation scenario. To be productive in SME environments, industrial robots must be easily configurable and quick to instruct to small volume production processes. Reliability and error handling are key issues in this context and will be addressed by introducing situation assessment on robot skill level. By this approach it will be possible to detect deviations from predicted or learned relations in execution of the skills as well as possibly predict future critical situations early to avoid failure. Cooperation with the operator has been discussed slightly within this thesis, but will also be investigated in much larger detail in the SMErobotics project.

Bibliography

J. S. Albus, H. G. McCain, and R. Lumia. *NASA/NBS Standard Reference Model for telerobot control system architecture (NASREM)*. U.S. Dept. of Commerce, National Institute of Standards and Technology, Gaithersburg, MD :, 1989 ed. edition, 1989.

J. Andersen, N. Andersen, and O. Ravn. *Vision Assisted Laser Scanner Navigation for Autonomous Robots*, volume 39, pages 111–120. Springer, 2006a. ISBN 978-3-540-77456-3.

J. Andersen, M. Blas, N. Andersen, O. Ravn, and M. Blanke. Traversable terrain classification for outdoor autonomous robots using single 2d laser scans. *Integrated Computer-Aided Engineering*, 13(3):223–232, 2006b.

J. Andersen, O. Ravn, and N. Andersen. Autonomous rule based robot navigation in orchards. In *7th IFAC Symposium on Intelligent Autonomous Vehicles*. International Federation of Automatic Control, 2010.

J. C. Andersen. *Mobile Robot Navigation*. PhD thesis, Technical University of Denmark, Department of Electrical Engineering, 2005.

N. A. Andersen and O. Ravn. Smr-cl, a real-time control language for mobile robots. In *in proc. of CIGR International Conference*, Beijing, oct. 11-14 2004.

V. Andronache and M. Scheutz. ADE - a tool for the development of distributed architectures for virtual and robotic agents. In *Proceedings of the Fourth International Symposium, "From Agent Theory to Agent Implementation"*, 2004.

B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *In PODS*, pages 1–16, 2002.

J. Barwise and J. Perry. Situations and attitudes. *The Journal of Philosophy*, 78(11):pp. 668–691, 1981. ISSN 0022362X. URL `http://www.jstor.org/stable/2026578`.

J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press, 1983. ISBN 0-262-02189-7.

M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media*, 2009. URL `http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154`.

A. B. Beck. Mission management for mobile robots. Master's thesis, Techincal University of Denmark, 2009.

A. B. Beck and N. A. Andersen. Robot hardware abstraction layer. Technical report, Technical University of Denmark, 2008.

A. B. Beck, N. A. Andersen, and O. Ravn. Mission management for mobile robots. In L. Akdahl, editor, *The fourth Swedish Workshop on Autonomous Robots*, volume I, Vesterås, September 2009. Robotdalen, Robotdalen.

A. B. Beck, N. A. Andersen, J. C. Andersen, and O. Ravn. Mobotware - a plug-in based software framework for mobile robots. In *7th IFAC Symposium on Intelligent Autonomous Vehicles*, Lecce, Italy, 2010.

A. B. Beck, C. Risager, N. A. Andersen, and O. Ravn. Spacio-temporal situation assessment for mobile robots. In *14th International Conference on Information Fusion*, Chicago, 2011.

J. Beringer and E. Hüllermeier. Online-clustering of parallel data streams. *Data and Knowledge Engineering*, 52:180–204, 2006.

R. Bischoff, T. Guhl, E. Prassler, W. Nowak, G. K. Kraetzschmar, H. Bruyninckx, P. Soetens, M. Hägele, A. Pott, P. Breedveld, J. Broenink, D. Brugali, and N. Tomatis. Brics - best practice in robotics. In *Proceedings for the joint conference of ISR 2010 (41st Internationel Symposium on Robotics) und ROBOTIK 2010 (6th German Conference on Robotics)*, pages 1–8, Munich, Germany - Parallel to AUTOMATICA, 7-9 June 2010. VDE Verlag.

C. M. Bishop. *Pattern recognition and Machine Learning*. Springer, 2006.

H. Bruyninckx. Open robot control software: The orocos project. In *Proc. of IEEE Int. conf. on Robotics and Automation*, 2001.

H. Bruyninckx. Real-time and embedded guide. Technical report, K.U.Leuven, Leuven, 2002.

J. Butterfield. Review - situation and attitudes. *The Philosophical Quarterly*, 36(143):pp. 292–296, 1986. ISSN 00318094. URL `http://www.jstor.org/stable/2219775`.

C. Coté, Y. Brosseau, D. Letourneau, C. Raievsky, and F. Michaud. Robotic software integration using marie. *International Journal of Advanced Robotic Systems*, 3(1), 2008.

R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010. URL `http://www.programmingvision.com/rosen_diankov_thesis.pdf`.

R. Diankov and J. Kuffner. Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Pittsburgh, PA, July 2008.

P. Duda, M. Jaworski, and L. Pietruczuk. On pre-processing algorithms for data stream. In *Proceedings of the 11th international conference on Artificial Intelligence and Soft Computing - Volume Part II*, ICAISC'12, pages 56–63, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-29349-8. doi: 10.1007/978-3-642-29350-4_7. URL `http://dx.doi.org/10.1007/978-3-642-29350-4_7`.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. Wiley, New York, 2001.

M. Dunham, Y. Meng, and J. Huang. Extensible markov model. In *Fourth IEEE International Conference on Data Mining*, pages 371–374, 2004.

N. Elkmann, J. Hortig, and J. Fritzsche. Cleaning automation. In *Handbook of Automation*. Springer, 2009.

M. Endsley. Towards a theory of situation awareness in dynamic systems. *Human Factors*, 37:32–64, 1995.

M. R. Endsley. Automation and situation awareness. In L. Erlbaum, editor, *Automation and human performance: Theory and applications*, pages pp. 163–181. Lawrence Erlbaum, 2006.

M. R. Endsley. Bringing cognitive engineering to the information fusion problem: Creating systems that understand situations. In S. T. Inc., editor, *Keynote at the 14th International Conference on Information Fusion*, Chicago, July 2011.

Fraunhofer IPA. Service robots in nursing homes - popular among residents and carers alike. PRESS RELEASE, July 2011. URL `http://www.care-o-bot.de/Produktblaetter/WiMi-Care_Serviceroboter_eng.pdf`.

D. Garagic, B. J. Rhodes, N. A. Bomberger, and M. Zandipour. Categorization of maritime anomalies for notification and alerting purpose. In *NATO Workshop on Data Fusion and Anomaly Detection for Maritime Situational Awareness*, La Spezia, Italy, 15-17 September, 2009.

B. Gates. A robot in every home. *Scientific American*, X:58–65, 2007.

B. P. Gerkey, R. T. Vaughan, K. Støy, A. Howard, G. S. Sukhatme, and M. J. Mataric. Most valuable player: A robot device server for distributed control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001)*, pages 1226–1231, Hawaii, October 29 - November 3 2001.

D. Goldberg and M. J. Mataric. Augmented markov models. Technical report, Computer Science Department, University of Southern California, 1999.

G. Gross, R. Nagi, and K. Sambhoos. Soft information, dirty graphs and uncertainty representation/processing for situation understanding. In *Information Fusion (FUSION), 2010 13th Conference on*, pages 1 –8, july 2010.

S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *41st Annual Symposium on Foundations of Computer Science (FOCS 2000)*, pages 359–366. IEEE, 2000.

D. Hall and J. Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6 –23, jan 1997. ISSN 0018-9219. doi: 10.1109/5.554205.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, Volume 11, 2009.

I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, Jan. 2000. ISSN 1077-2626. doi: 10.1109/2945.841119. URL `http://dx.doi.org/10.1109/2945.841119`.

IFR Statistical Department. World robotics 2012. Technical report, International Federation of Robotics, 2012.

International Frederation of Robotics. Service robots. Website, August 2012. URL `http://www.ifr.org/service-robots/`.

R. N. Jørgensen, M. Nørremark, C. Sørensen, and N. A. Andersen. Utilising scripting language for unmanned and automated guided vehicles operating within row crops. *Computers and Electronics in Agriculture*, 62(2):190–203, 2008. ISSN 0168-1699.

G. A. Klein. Recognition-primed decisions. *Advances in man-machine systems research*, pages p. 47–92, 1989.

J. Kramer and M. Scheutz. Development environments for autonomous mobile robots: A survey. *Autonomous Robots*, 22(2):101–132, Feb. 2007. ISSN 0929-5593. doi: 10.1007/s10514-006-9013-8. URL `http://dx.doi.org/10.1007/s10514-006-9013-8`.

R. Lane, D. Nevell, S. Hayward, and T. Beaney. Maritime anomaly detection and threat assessment. In *Information Fusion (FUSION), 2010 13th Conference on*, pages 1 –8, july 2010.

E. Little and G. Rogova. Ontology meta-model for building a situational picture of catastrophic events. In *Information Fusion, 2005 8th International Conference on*, volume 1, page 8 pp., july 2005. doi: 10.1109/ICIF.2005.1591935.

J. Llinas, C. Bowman, G. Rogova, A. Steinberg, E. Waltz, and F. White. Revisiting the jdl data fusion model ii. In *In P. Svensson and J. Schubert (Eds.), Proceedings of the Seventh International Conference on Information Fusion (FUSION 2004*, pages 1218–1230, 2004.

D. MacKenzie and R. Arkin. Evaluating the usability of robot programming toolsets. *The International Journal of Robotics Research*, 17 - 4:381–401, 1998.

A. Makarenko, A. Brooks, and T. Kaupp. On the benefits of making robotic software frameworks thin. In E. Prassler, K. Nilsson, and A. Shakhimardanov, editors, *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'07) Workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware*, November 2007. URL `http://www.cas.edu.au/download.php/makarenko07_iros_benefits_of_thin_frameworks.pdf?id=1748`.

P. Mantegazza, E. Bianchi, and L. Dozioi. Rtai: Real-time application interface. *Linux Journal*, Issue 72:p. 142–150, 2000.

G. McLachlan and K. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, 1988.

G. Metta, P. Fitzpatrick, and L. Natale. Yarp: Yet another robot platform. *International Journal of Advanced Robotic Systems*, 3(1):43–48, 2006. ISSN 1729-8806.

D. Meyer-Delius, C. Plagemann, G. von Wichert, W. Feiten, G. Lawitzky, and W. Burgard. A probabilistic relational model for characterizing situations in dynamic multi-agent systems. In *In Proc. of the 31th Annual Conference of the German Classification Society on Data Analysis, Machine Learning, and Applications (GFKL)*, Freiburg, Germany, 2007.

D. Meyer-Delius, C. Plagemann, and W. Burgard. Probabilistic situation recognition for vehicular traffic scenarios. In *2009 IEEE International Conference on Robotics and Automation*, pages 459–464, Kobe, Japan, May 12-17 2009a.

D. Meyer-Delius, J. Sturm, and W. Burgard. Regression-based online situation recognition for vehicular traffic scenarios. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, USA, 2009b.

M. Montemerlo, N. Roy, and S. Thrun. Perspectives on standardization in mobile robot programming: The carnegie mellon navigation (carmen) toolkit. In *Proc. Of the IEEE/RSJ Int. Conf. On Intelligent Robots and Systems*, Las Vegas, Nevada,, 2003.

I. A. Nesnas, R. Simmons, D. Gaines, C. Kunz, A. Diaz-Calderon, T. Estlin, R. Madison, J. Guineau, M. McHenry, I.-H. Shu, and D. Apfelbaum. Claraty: Challenges and steps toward reusable robotic software. *International Journal of Advanced Robotic Systems*, Vol. 3:pp. 023–030, 2006.

L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani. Streaming-data algorithms for high-quality clustering. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 685 –694, 2002. doi: 10.1109/ICDE.2002.994785.

T. C. Pauchant and I. I. Mitroff. *Transforming the Crisis Prone Organizatio*. Jossey-Bass Publishers, 1992.

I. Paul. Google's self-driving car licensed to hit nevada streets. PC-World Website, May 8 2012. URL `http://www.pcworld.com/article/255204/googles_selfdriving_car_licensed_to_hit_nevada_streets.html`.

M. Prentice, M. Kandefer, and S. Shapiro. Tractor: A framework for soft information fusion. In *Information Fusion (FUSION), 2010 13th Conference on*, pages 1 –8, july 2010.

M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *International Conference on Robotics and Automation*, Open-Source Software workshop, 2009. URL `http://robotics.stanford.edu/~ang/papers/icraoss09-ROS.pdf`.

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77, Issue 2:257–286, 1989.

R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In V. Lifshitz, editor, *Artificial intelligence and mathematical theory of computation: papers in honour of John McCarthy*, pages Pages 359–380, San Diego, CA, USA, 1991.

B. Ristic, B. L. Scala, M. Morelande, and N. Gordon. Statistical analysis of motion patterns in ais data: Anomaly detection and motion prediction. In *11th International Conference on Information Fusion*, Cologne, Germany, July 2008.

G. Rogova. Higher level fusion and decision support for situation management : Challenges and computational approaches. In *Toturial at FUSION 2011-The Fourteens International Conference on Information Fusion*, Chicago, IL, 5 July 2011 2011.

J. Roy. Automated reasoning for maritime anomaly detection. In *NATO Workshop on Data Fusion and Anomaly Detection for Maritime Situational Awareness*, La Spezia, Italy, 15-17 September, 2009.

J. Roy and M. Davenport. Categorization of maritime anomalies for notification and alerting purpose. In *NATO Workshop on Data Fusion and Anomaly Detection for Maritime Situational Awareness*, La Spezia, Italy, 15-17 September, 2009.

S. J. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010. ISBN 978-0-13-207148-2.

S. Schwoegler, S. Blackman, J. Holsopple, and M. Hirsch. On the application of multiple hypothesis tracking to the cyber domain. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1 –6, july 2011.

G. Shroff, S. Sharma, P. Agarwal, and S. Bhat. A blackboard architecture for data-intensive information fusion using locality-sensitive hashing. In *Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on*, pages 1 –8, july 2011.

SICK. Nav3xx - laser measurement technology. https://www.mysick.com/saqqara/im0044787.pdf, May 2012.

C. Stachniss, U. Frese, and G. Grisetti. Openslam. Website, 2010. URL `www.openslam.org`.

A. N. Steinberg, C. L. Bowman, and F. E. White. Revisions to the JDL data fusion model. In B. V. Dasarathy, editor, *Sensor Fusion: Architectures, Algorithms, and Applications III*, volume 3719, pages 430–441. SPIE, 1999. URL `http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal&id=PSISDG003719000001000430000001&idtype=cvips&gifs=yes`.

M. Sudit, M. Holende, and A. Stotz. Inferd and entropy for situational awareness. *Journal of Advances in Information Fusion*, Volume 2(Number 1):3–21, June 2007.

N. N. Taleb. *Fooled by Randomness: The Hidden Role of Chance in the Markets and in Life*. W. W. Norton & Company, 1st edition, Oct. 2001. ISBN 1587990717. URL `http://www.worldcat.org/isbn/1587990717`.

N. N. Taleb. *The Black Swan. The Impact of the Highly Improbable*. Random House Inc., 2008. ISBN 0812979184.

P. Tjell and S. Hansen. Laser-based navigation in orchard. Master's thesis, Technical University of Denmark, 2008.

R. T. Vaughan. Massively multi-robot simulations in stage. *Swarm Intelligence*, 2(2-4):189–208, 2008.

M. Viager. Flexible mission execution for mobile robots. Individual course, Technical University of Denmark, Department of Electrical Engineering, 2012. Advisor(s): Jens Christian Andersen, Nils Axel Andersen, Anders Billesø Beck.

F. E. White. Data fusion lexicon. Technical report, Joint Directors of Laboratories, Technical Panel for C3, Data Fusion Sub-Panel, Naval Ocean Systems Center, San Diego, 1987.

T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. In *International Conference on Management of Data*, pages 103 – 114. ACM New York, NY, USA, 1996.

www.elektro.dtu.dk

Department of Electrical Engineering
Automation and Control
Technical University of Denmark
Elektrovej
Building 326
DK-2800 Kgs. Lyngby
Denmark
Tel: (+45) 45 25 35 76
Fax: (+45) 45 88 12 95
Email: info@elektro.dtu.dk

ISBN 978-87-92465-64-1